Interface
2003 February

[表紙デザイン：㈱プランニング・ロケッツ]

Contents 2003 February

特集

無線による高速データ伝送が身近になってきた！

Interface

話題のテクノロジ解説

ショウレポート&コラム

一般解説&連載

情報のページ

組み込み技術の総合展示会

Embedded Technology 2002

北村俊之

「あなたが求める『組込み技術』が必ず見つかる一週間！」をテーマに「Embedded Technology 2002」が11月20日(水)～22(金)の3日間，パシフィコ横浜で開催された(**写真1**)．主催は，(社)日本システムハウス協会．名称を「MST」から「Embedded Technology」に変更した同展示会は，今年で通算16回目を迎え，出展社数270団体，カンファレンス総数120セッションと規模を大幅に拡大しての開催となった．

〔**写真1**〕**入場口のようす**

〔**写真2**〕**ユニバーシティパビリオン**

昨年に引き続き，「インドパビリオン」をはじめ海外からの参加も増大し，大学研究室の出展コーナーである「ユニバーシティパビリオン」(**写真2**)も充実が図られていた．また，「システムハウス・SIパビリオン」，「テスト＆メジャメントゾーン」，「プラットホームゾーン」など，時代のニーズに応えた出展企画も多く見られた．展示会の前日2日間には，12コース設けられたチュートリアルも開催され，今回はじめての試みとして注目を集めていた．最終的な延べ来場者数は，3日間で16,741人に達した．

● 注目される展示物

先頃，タブレットPCの発表で話題を集めたマイクロソフトは，「Windows CE.NET 4.1」および「Windows XP Embedded」を中心に展示を行っていた．同社のパートナーパビリオンでは，10社以上のパートナー企業がEmbedded Windowsを利用した多彩なソリューションを展示しており，注目を集めていた(**写真3**)．

〔**写真3**〕**マイクロソフトのパートナーパビリオン**

エンジニリアリングサポートを含むプログラマブルシステムソリューションを提供するザイリンクスは，「Virtex-Ⅱ Pro」，「Spartan-Ⅱ E」，「ISE5.1i」，「CoolRunner-Ⅱ」などのFPGA，ソフトウェア開発ツールなどの展示を行っていた．とくに，開発システムである「ISE5.1i」は，来場者の注目が高いとのことであった．

ウィンドリバーは，リアルタイムOS「VxWORKS」や統合開発環境「Tornado」，インテリジェントデバイスのソフトウェア開発を支援するプラットホームなどを展示していた．

リアルタイムOSや多彩なミドルウェアを展示していたのがグレープシステムである．「ThreadX-μITRON」はCISCで最小約2.5Kバイトのコンパクトな RTOSで，同等クラスでは業界No.1の速度を誇るという．また，同社ではT-Engine対応のUSB，Bluetooth，FTP，HTTPなどのミドルウェアも注目度が高いという．「GR-USB/OTG」は，組み込みシステム用のUSB OTG機能を提供するものだ．

〔**写真4**〕**車載情報システムの例**

富士通は，Bluetooth，ブロードバンドルータ，マイクロコントローラなどを中心に展示を行っていた．ブロードバンドルータソリューションは，双方向100MHzフルレート転送を実現するホームブロードバンドルータ向けIPパケット高速処理LSIの紹介を行っていた．また，SoC開発環境では，UMLとC言語をベースとしたシステムLSI設計手法の紹介を行っていた．

T-Engineプロジェクトに早い時期から参加していたパーソナルメディアは，日立製作所，三菱電機のCPUを利用したT-Engine開発キットの新製品やソリューションを多数展示していた．ITS Japanおよび自動車走行電子技術協会は，テレマティクスサービスやプローブ情報システム，インターネットITSシステムなど最新車載技術の展示を行っており，来場者の注目を集めていた(**写真4**)．日本テキサスインスツルメンツは，DSPの新製品，製品のロードマップ，開発ツールなどを中心に展示を行っていた．WLAN，ディジタルスチルカメラ向けのソリューションなどのデモも行っており，パートナー企業19社によるソリューションの展示も注目を集めていた．

〔**写真5**〕**Metrowerks Embedix SDKの展示**

開発ツール「CodeWarrior」を中心とした開発ソフト，ハードウェアおよび総合的なサービスを提供するメトロワークスは，「Symbian OS」，「Tao intent」をはじめとするJavaやEmbedded Linuxなどの最新テクノロジを展示していた(**写真5**)．イーソルはコンフィギュレータ，ビルダ，デバッグテストツール，品質向上テストツールなどの機能を搭載する「eBinder」を中心とした展示を行っていた．

〔**写真6**〕**QNX Momentics開発スイートの展示**

QNXソフトウェアシステムズは，MIPS，ARMなど多様なターゲットに対応し，開発者にあった言語と開発ホストを使用できる「QNX Momentics開発スイート」の展示を行っていた(**写真6**)．

アームは，最新の「RealView」開発ツールの概要をデモを交えながら行っていた．同製品はSoC設計やSoC統合に向けた統合的なツールであり，ファーストシリコンの信頼性の向上と製品化までの期間を大幅に短縮するとのことである．こちらのブースでも20社近くのパートナー企業が，ARMテクノロジを利用したソリューションの展示を行っていた(**写真7**)．

〔**写真7**〕**RealViewによる開発を展示していたARM**

サイバネットシステムは，テキサスインスツルメンツ社製DSPに対するトップダウン設計ツールである「Embedded Target for Texas Instrument C6000 DSP」および「MATLAB Link for Code Composer Studio Development Tools」を紹介し，実機デモを行っていた(**写真8**)．

〔**写真8**〕**DSPデザインとMATLABの連携に力を入れるサイバネット**

Show report

家電機器をもネットワーク化するインフラ

Universal Plug and Play Asia Summit in TOKYO

SUGATECH

北米以外では初の開催となるUPnP Asia Summit in TOKYOが，11月11日と12日の両日，東京フォーラムで開催された．これまでインテルのIDF-JやマイクロソフトのPlugfest内でUPnPに関する技術情報が公開されてはいたが，UPnPフォーラムのメンバーが開発者を対象として日本国内で動向を紹介するのは初めてのことであり，235名の参加者が会場に集まった．

● 基調講演

UPnP運営委員会のメンバーであるマイクロソフトの加治佐氏の基調講演に始まり，技術情報以外にも，フォーラムにおける現状の活動内容や今後の方針が説明された（**写真1**）．現在日本から参加し，各運営委員会の主要メンバーとなっているソニー，キヤノン，三菱電機，リコーの4社からはそれぞれの運営委員会の状況が紹介され，製品デモも行われた．

〔写真1〕マイクロソフトの加治佐氏

〔写真2〕Linksysのワイヤレスアクセスポイントルータ

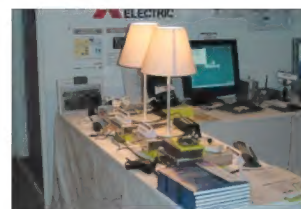
● UPnPの動向

UPnP自身がめざすところは，パソコン周辺機器やコンシューマ機器をネットワークに接続し，お互いに情報を共有して制御を可能とすることだが，ネットワーク上で機器の接続と制御を容易にするための仕様書（UPnP Version 1）が策定された．仕様書は運営委員会ごとに発行され，最初の仕様書としては，2001年11月にインターネットゲートウェイ運営委員会によるものがある．その結果として，各社のブロードバンドルータがUPnP対応を始めている（**写真2**）．

イメージング（プリンタとスキャナ），オーディオ/ビデオの仕様書は2002年夏から秋にかけて策定を完了しているので，今後各社から対応機器の発表が期待される．もともとPCの周辺機器であるルータやプリンタそしてスキャナなどがUPnPを導入することは当然の進歩といえるが，UPnPはさらにコンシューマ機器との融合をめざしている．そのため，UPnP Audio/VideoやUPnP Home Automationといった運営委員会が発足している．

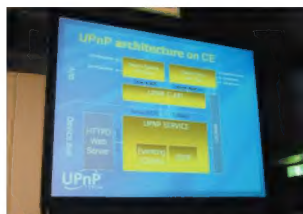
ソニーによるデモでは，VAIO上のコンテンツを，VAIOルームリンクに接続されたテレビにリモコンからコンテンツ配信の制御を行っていた（**写真3**）．UPnP AV（オーディオ/ビデオ）の仕様ではコンテンツを配信する側をメディアサーバ，再生機能をもつデバイスをレンダラと呼び，リモコンにあたるコントロールポイントが二つのデバイスの制御を行う．この分野の製品としてレンダラとコントロールポイントが一つになった再生デバイスが普及すると予測される．UPnP AV仕様ではストリーミングのプロトコルとデータフォーマットは規定せずに互換性のチェックのみを行うため，デバイス開発者は独自のストリーミングプロトコルやデータフォーマットを採用することができる．

〔写真3〕VAIOを使ったソニーのデモ

UPnP Home Automation運営委員会では，ホームオートメーションに関するDCP（Device Control Protocol）の策定を行っており，照明機器や防犯カメラ/システム，電力システムなどの制御やモニタをUPnPによって実現することをめざしている．三菱は電力線を利用したPLC（Power Line Control）ネットワークとUPnPネットワークを，イスラエルのITRAN社が開発したIT800モデムによって結合し，コントロールポイントであるPocket PCで制御するデモを行っていた（**写真4**）．PLCネットワークに接続される照明機器などのデバイスはIPアドレスをもっていないため，実際の制御はPLCモデム間で行われる．すでに家庭内に引き回されている電力線を，インフラとしてそのまま利用できるのが最大のメリットである．

〔写真4〕三菱電機の電力線によるPLCネットワーク

また，METROLINK社では，家電製品をUPnP TCP/IPネットワークで簡単に結合するソリューションを提示していた．三菱との共同開発によるM16C評価ボードに同社のIPWorks OSを搭載することで，完全なUPnPインフラを必要としないピアツーピアでの機器接続が可能であるという．**写真5**のデモはCD-ROMプレーヤで再生した音楽を，UPnP経由で接続されたスピーカで再生するというもので，独立したコントロールポイントを必要としないことが特徴だ．

〔写真5〕M16Cボードを使用したMETROLINK社のデモ

今回のフォーラムでは，各社が提供しているUPnPの開発環境の説明も行われた（**写真6**）．米国インテルからは，インテルUPnPコアSDKやUPnP対応の技術ツール，米国マイクロソフトからはWindows CE .NET SDKなどがデモを交えて紹介された．Windows CE上ではデバイスをホストとするAPI，コントロールポイントAPIそしてブリッジ機能がサポートされるため，PlatformBuilderを使用してアプリケーションからデバイスまでの開発が可能であるという．

〔写真6〕UPNPのWindows CEにおける対応

また，日本国内での一般フォーラムとしては異例の長さの質疑応答が行われた（**写真7**）．何点かの回答を紹介すると，UPnP V2の策定終了は2004年以降となるので，これから開発するメーカーはV1仕様で製品化すべきであるということ．V2ではIPv6のサポートが要求されるが，IPv4の利用を可能とする自動トンネリングがサポートされるという．

〔写真7〕質疑応答のようす

Show&News Digest

バージョンアップ機能をもった μITRON仕様OSの開発を開始

■日時：2002年11月20日(水)
■場所：パシフィコ横浜(神奈川県横浜市)

Embedded Technology 2002でのプレス発表において，(株)エーアイコーポレーションとTOPPERS/IDLプロジェクトチームにより，バージョンアップ機能をもったμITRON仕様OSの開発が開始されることが発表された．

従来のμITRON仕様では，モジュールを静的にリンクしておく必要があり，ソフトウェアのバージョンアップなどを行うためには工場などで全体更新を行うか，2倍のメモリを用意して通信で入れ替えるなどの方法が必要だった．

今回発表されたIDL(μITRON with Dynamic Loading)は，動的リンクを行う際にサーバの存在を前提とし，サーバ上でリロケート作業を行い，作成したモジュールを機器にダウンロードするというものである．

質疑応答では，機器のメモリマップなどの情報をサーバへ送信することに対する不安の声も上がったが，送信データ暗号化などの機能は機器の別のレイヤで実現しているため，IDLそのものには盛り込まなかったとのことだった．

エーアイコーポレーションの加藤博之氏

Phil Pompa氏に AMD Alchemyソリューションを聞く

■日時：2002年11月20日(水)
■場所：パシフィコ横浜(神奈川県横浜市)

Embedded Technology 2002の会場で，AMDのVice President of MarketingであるPhil Pompa氏に，同社の組み込み向けソリューションについてインタビューを行った．

編　今回発表されたAlchemyソリューションについて教えてほしい．

Phil Pompa　AlchemyはMIPS32をコアとし，SDRAMコントローラ/PCI/USB/Ethernet/IrDA/AC97/UART/LCDコントローラなどを1チップにまとめたソリューションだ．内蔵周辺機能の種類により，Au1000/Au1100/Au1500の3種類のラインナップがあり，どれも最高500MHzで動作する．

編　組み込みCPUとしてはかなり豊富な機能を搭載しているが，消費電力や価格の不安はないのか？

P　各周辺機能ブロックは，それぞれ独立にクロックの供給を止めることができる．また，周辺機能製品の有無によりラインナップを増やすことはせず，3種類のみにしたことにより，チップのコストダウンにつながっている．

編　同時発表のワイヤレスLANソリューションについても教えてほしい．

P　Am1772は，2チップ構成の802.11b対応ワイヤレスLANチップセットだ．ホストI/FがDMAに対応しているほか，チップ構成を見直し，A-D/D-Aを前段に移すことにより，アナログ信号を長距離引き回すことがなくなった．そのため，デザインの自由度が増している．

編　今後のロードマップを教えてほしい．

P　AlchemyとAm1772を1チップに統合したソリューションを提供する予定だ．

Phil Pompa氏

RadiSys, Ron Dilbeck氏への インタビュー

■日時：2002年11月8日(金)

インテル系の組み込み用途向けボードコンピュータなどを開発・販売しているラディシス・コーポレーション(RadiSysCorp. http://www.radisys.com/)のChief Operating Officer(COO)であるRon Dilbeck氏にインタビューを行った．ラディシスはリアルタイムOS「OS-9」の開発ベンダーであるマイクロウェア・システムズを2001年に買収した．

編　御社について簡単に紹介してほしい．

Dilbeck　もともと創業者がインテルx86シリーズのチーフエンジニア出身ということもあり，インテル系のハードに強く，長期供給保証/高信頼性のボードコンピュータを提供している会社だ．

編　マイクロウェア買収の経緯と，その成果などは？

D　買収以前に，マイクロウェアのOS-9がIXP1200ネットワークプロセッサをサポートするとき，一緒に作業した．同社が買収先を探していると聞いて手を上げ，互いに条件が折り合った．買収による効果としては，顧客の幅が広がったことがあげられる．ボードビジネスでは新規ながら，すでにOS-9の顧客だった会社からハードの新規受注を受けるなど，良い効果が出てきた．日本のメーカーは性能や信頼性などに対する要求が厳しいが，医療分野などでの組み込み機器市場の成長が見込めると期待している．

Ron Dilbeck氏

フジワラヒロタツの現場検証(67)

周期

寒い日が続いていますが，風邪などお召しになっていませんでしょうか．凍てついた季節，まどろみの布団から抜け出して出社するのは本当に辛いものです．しかも仕事による慢性的な睡眠不足に加え，ストレス発散のために時折飲みに出ちゃったりするものですから，なおさら自分の首を絞めています(夜になると目が冴えるものですから，ついつい朝の事情を忘れてしまうというわけで……)．

季節も，日本経済も，ついでに自分の懐も冬の時代．せめて仕事だけは順調に……と思うのですが，やはりそうはいきません．うまく動かないプログラムを前に，思わず嘆息してしまいます．筆者は近頃でこそ物事を前向きに考えるようになってきましたが，SF好きでマンガ描きであり，コンピュータもハードじゃなくてソフトを選んだことからもおわかりになるように(?!)，悪いほうに物事を考えるほうが得意です．

冬，寒さに身をさらして鉄道のレールを凝視しつつ通勤電車を待っていたりすると，思わずそういった「本性」がむくむくと身をもたげてきます．

こんなにデバッグが進まないのはやはり才能がないからなのだ，やはりこの仕事は向いていないのだと，何もかも嫌になって，転職する自分を空想していきます．ハンバーガーショップでひたすらハンバーガーをつくる自分，古本屋で働く自分，年金暮らしをする自分(さすがに年金をもらえるのはまだまだ先の話)．なんてすばらしい自分でしょう！　もうそこにはバグ表も，単体テストも，なぜかいつも線表から二日遅れになる進捗報告すらないのです!!

こういった「へこむ」時期は，筆者にとって周期的に訪れるようで，そのたびに自らの才能のなさと適応できない職業生活を思って絶望的な気持ちになります．青雲の志に燃えて購入したソフトウェア工学の冊子群は「序」と「日本語版に向けての序」を読んだだけで日経＊＊の下に埋もれていますし，目の前のデバッグに追われつつの食べ物は，カップラーメンとカップうどん，牛丼とカツ丼，カレーライスとライスカレーばかりです．

ソフトウェアのテスト工程の打ち切り方の一つに，テストで出たバグの個数をプロットしていき，その個数がある程度落ち着いてきたならば品質が安定したと見てテストを終えるという考え方があります．はて，筆者の品質は安定しているのでしょうか？　いつも慣れたあたりでほかの分野に手を出して，初期不良が多いテスティング初期のバグ件数をキープしたまま，安定期にはほど遠い職業生活を繰り返すのではありますまいか．

しかし最近，良いストレス解消法を見つけました．夜中にこっそり，自宅の皿洗いをするのです．皿洗いはやればやっただけ，確実に成果が上がります．決してデバッグのように後戻りしないのです．これは素晴らしいことで，デバッグではとうてい得られない達成感を獲得することが，才能のないプログラマ風情にも存分に味わえるのです！　さらには，コビトさんがきたコビトさんがきたと家人が喜んでくれるという，これまたプログラマには滅多にない，感謝まで味わえることも確実なのです．

藤原弘達　(株)JFP　デバイスドライバエンジニア，漫画家

無線による高速データ伝送が身近になってきた！

ワイヤレスネットワーク技術入門

最近，各種無線LANやBluetoothなどのワイヤレスネットワークが，手軽にネットワークを構築/拡張する手段として注目を集めている．そこでまず，現在もっとも普及しているIEEE802.11方式の無線LANの技術や標準化動向などを中心に，無線LANの基礎から標準化動向までを解説する．

次に，Linux上で動作するBluetoothプロトコルスタックとその評価ボードに関して，技術的詳細と使用例を解説し，実際にBluetooth対応製品を開発したときの問題点も考察する．

そして，無線LANの標準規格IEEE802.11aで変調方式として使われるOFDMについて基礎原理を説明し，OFDMデータモデムの設計事例を紹介しながら，開発のポイントをくわしく解説する．

また，100Mbps以上のデータ伝送を実現する，60GHz帯の電波を使った無線伝送技術の基礎と，課題となる周波数安定性の問題を解決するための「ミリ波自己ヘテロダイン伝送方式」について説明する．最後に，前述した60GHz帯の電波を使うのとは違うアプローチ，つまり搬送波を用いずに1ns以下の非常に時間幅の短いパルスを用いて通信を行い，100Mbps以上の伝送速度を実現する通信方式「UWB」に関して，技術的課題と法制度化を含む今後の発展性について基礎から解説する．

IEEE802.11方式を中心とした

Chapter 1

ワイヤレスネットワーク技術の現況

● 阪田 徹

オフィスはもとより，家庭内においても情報化が進み，パソコンは一家に一台から，一人一台の時代となってきている．インターネットの急速な普及とパソコンや周辺機器の増加により，ネットワーク構築の要望は当然高まってきている．近年，各種無線LANやBluetoothなどのワイヤレスネットワークが，手軽にネットワークを構築・拡張する手段として注目を集めている．なかでも，2.4GHz帯の電波を用いた無線LAN機器は，コンシューマ向け製品の低価格化にともない爆発的な普及が進んでいる．本章では，現在もっとも普及しているIEEE802.11方式の無線LANの技術や標準化動向などを中心に，無線LANについて解説する．（筆者）

● 無線LANの特徴

無線LANは，ネットワークを構築する際にネックとなるケーブル敷設の問題を解決する有力な手段として普及してきた．さらに，ノートパソコンやPDAといった携帯型情報端末の普及により，端末を移動して使いたいといった要望に応えるものとして期待されている．

このように無線LANの特徴は，

- ケーブル敷設の煩雑さから開放されるため，既設住宅などへのLAN構築の自由度，拡張性を提供できる
- 通信媒体が無線であることから，端末の可搬性，ユーザーの移動性を損なわないで，通信エリア内のいたるところからネットワークへのアクセスを可能とする利便性を提供できる

という点にある．

前者は無線LANの原点ともいえるもので，「配線」の呪縛から解き放たれたいといった要望であるが，後者は，無線という特徴と携帯型端末の組み合わせにより，新たなサービスへの発展を望むものと考えられる（**図1**）．

● 無線LANの発展と普及

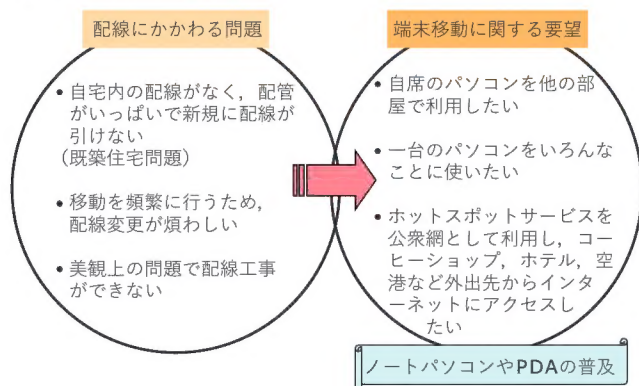
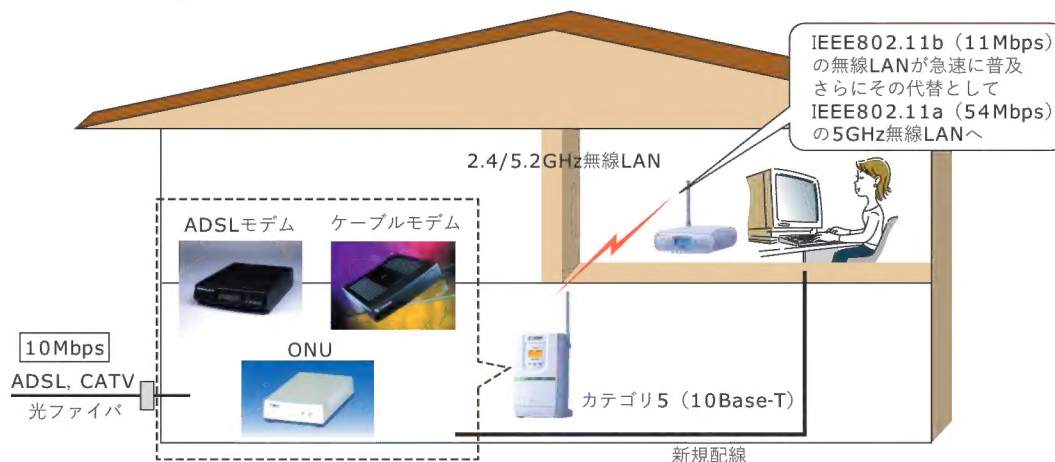
オフィスや家庭内に増加する情報機器を結ぶワイヤレスネットワークも，ブロードバンド化の波を受けて発展してきている．ISDNがおもなアクセスラインであり，伝送速度がkbpsレベルだった時代は，PHSや2.4GHz帯無線LAN（IEEE802.11，～2Mbps）を適用したワイヤレスTAなどがネットワークのワイヤレス化に利用されてきた．しかし，近年の爆発的なADSL，CATVなどの普及によるブロードバンド化にともない，ワイヤレスネットワークにも広帯域化が望まれていた．そこで，アクセスラインに対応した伝送速度をもつ2.4GHz帯無線LAN（IEEE802.11b，～11Mbps）が，1999年のアップルコンピュータ社のAirPortの発売をきっかけとした機器の低価格化もあいまって，広く普及してきている．

さらに2001年後半からは，より広帯域かつ他のシステムとの干渉のない5GHz帯を用いた無線LAN（IEEE802.11a，～54Mbps）が製品化され，ブロードバンド時代のワイヤレスネットワーク媒体として無線LANが市民権を得るにいたった（**図2**）．現在，今後来たるべき光ブロードバンドの時代に向けた広帯域伝送化（100Mbps級）を実現する超高速無線LANの検討が進められている．

● 公衆無線LANサービスへの展開

ノートパソコンの普及とともに，外出先でもブロードバンド環境を享受したいという要望が高まっている．家庭，オフィス，外出先などでシームレスなブロードバンド環境を実現するノマディックワイヤレスアクセス（NWA，**図3**）の一つの形として，家庭やオフィスで利用している無線LAN端末を適用した公衆無線LANサービスが，多くの事業者によってサービスが開始または検討されている．

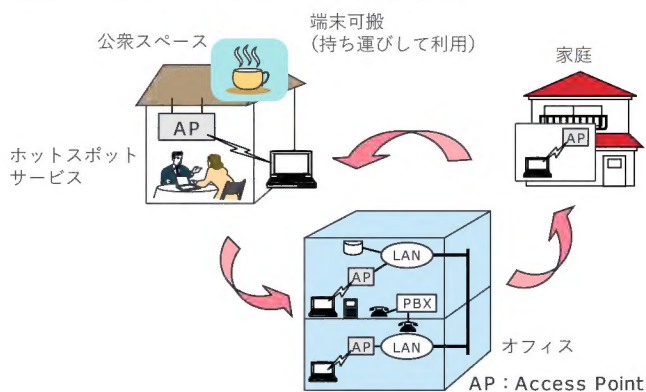
NTTコミュニケーションズでは，2002年5月より，ホテルのロビーやファーストフード店，コーヒーショップなどの公衆スポットでの無線LANサービス「ホットスポット」の商用サービスを開始した（**図4**）．2002年度内には，約1000箇所のアクセスポイントを設置する予定である．本サービスでは，もっとも一般的なIEEE802.11bに加え，より快適なブロードバンド環境を提供する

〔図1〕無線LAN導入のトリガ

〔図2〕高速アクセスライン(10Mbps)時代のホームネットワーク

ためIEEE802.11aとのデュアルバンド・アクセスポイントを導入し，世界初のデュアルバンド対応公衆無線LANサービスを展開している．無線LANは，こうした「ユビキタスサービス」を支えるアクセスネットワークの媒体の一つとして，期待されている．

1 無線LANの標準化を担うIEEE802.11ワーキンググループ

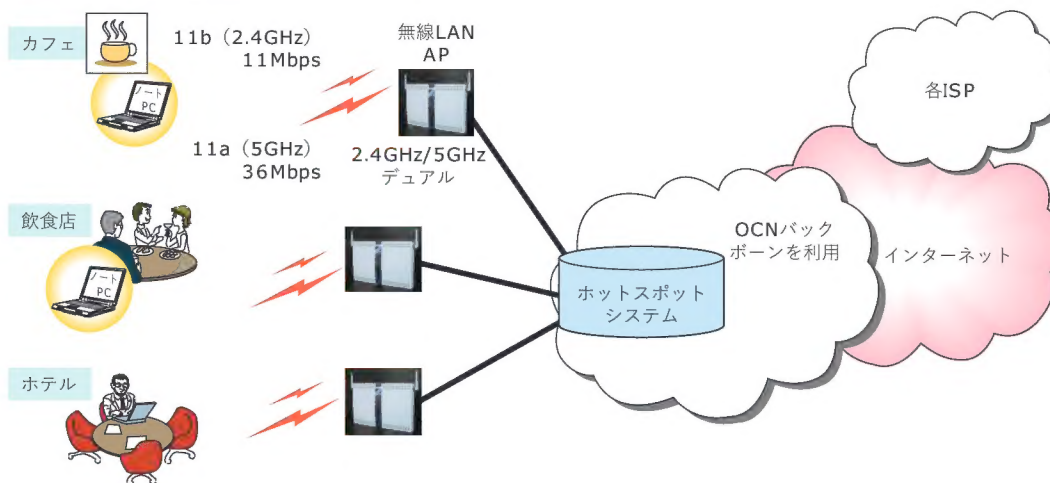
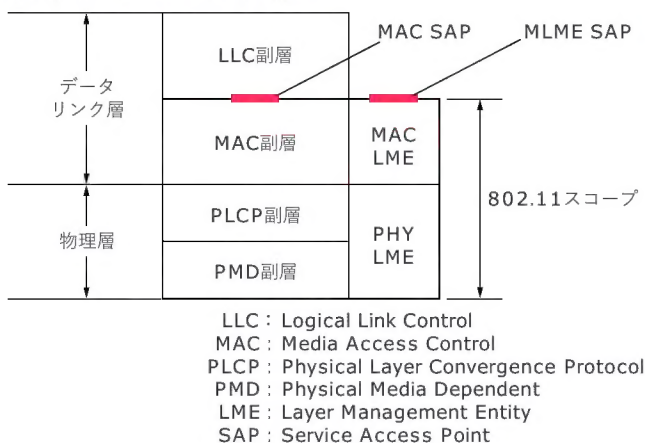
無線LANの標準化を行う代表的な機関は，米国のIEEE(Institute of Electrical and Electronics Engineers)の802委員会配下にあるワーキンググループ(WG)11である．このIEEE 802.11ワーキンググループは，1990年に設立されて以来，7年の年月を経て最初の標準規格「IEEE802.11」を完成させた．

この規格の対象となるのは，データリンク層の分散制御や集中制御などのプロトコルに関するMAC(Media Access Control，媒体アクセス制御)レイヤと，データ伝送速度や無線周波数帯域などに関する物理レイヤである．また各層のマネジメント機能も規定されている(**図5**)．

〔図3〕ノマディックワイヤレスアクセス(NWA)

MACレイヤの技術としては，Ethernetで用いられる自律分散制御方法を踏襲したCSMA/CA(Carrier Sense Multiple Access with Collision Avoidance，衝突回避機能付きキャリア感知多元接続)と，オプション規定としてポーリング方式が採用されている．また，物理レイヤには，

〔図4〕「ホットスポット」サービスの構成

〔図5〕IEEE802.11のレイヤ構造

LLC：Logical Link Control
MAC：Media Access Control
PLCP：Physical Layer Convergence Protocol
PMD：Physical Media Dependent
LME：Layer Management Entity
SAP：Service Access Point

〔表1〕IEEE802.11のおもな検討グループ構成（2002/10月現在）

Task Group	検討内容	状　況
TGa	5GHz WLAN（OFDM）	作業終了
TGb	2.4GHz WLAN（CCK）	作業終了
TGc	MAC Supplement to IEEE802.1d	作業終了
TGd	Update of Regulatory Domains → R-REG	進行中
TGe	MAC Enhancement（QoS Support）	進行中
TGf	IAPP（Inter-Access Point Protocol）	進行中
TGg	High Rate Extensions to 802.11b（over 20M bit/s）	進行中
TGh	Spectrum Management for 802.11a（TPC&DFS）	進行中
TGi	Enhanced Security	進行中
HT SG	High Throughput Study Group	進行中
WNG SC	Wireless Next Generation Standing Committee	進行中

① 2.4GHz帯を用いた直接拡散方式（DS-SS）
② 2.4GHz帯を用いた周波数ホッピング方式（FH-SS）
③ 赤外線通信方式

の3種類が規定されている．2.4GHz帯での方式は，ISM(Industrial, Scientific and Medical applications)バンドと呼ばれる周波数帯のため，耐干渉性に優れたスペクトル拡散方式が採用された．伝送速度は1Mbps，2Mbpsである．

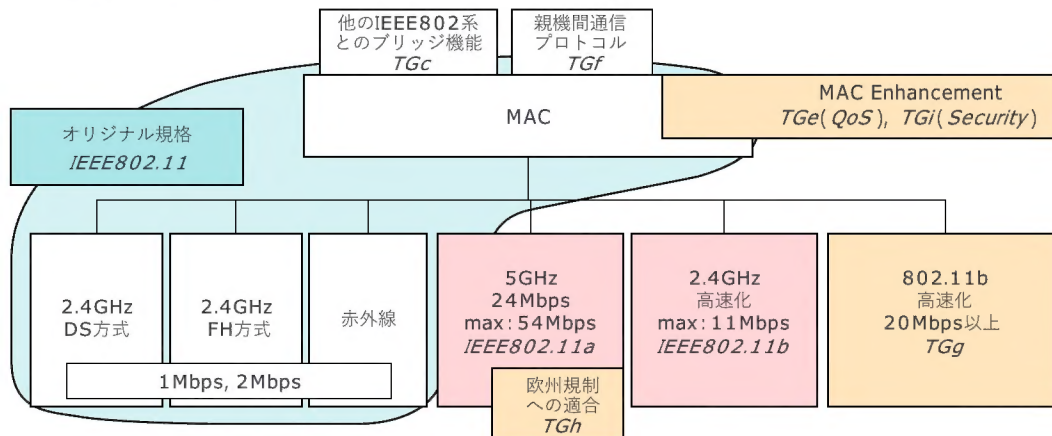
現在，IEEE802.11ワーキンググループでは，**表1**に示すようなタスクグループ（TG）に分かれて審議がなされている．各タスクグループは，設立された順に小文字のアルファベットが付与される．すでに標準化作業を終了したものもあるが，無線LANの高速化や高品質化を目的とした審議が続けられている．

各タスクグループの活動を，レイヤ構造に対応して整理したものを**図6**に示す．IEEE802.11ワーキンググループでは，「一つのMACレイヤに複数の物理レイヤ」を基本的なコンセプトとしている．これは，上位のIPレイヤから見てどの無線LANでも同一に扱えることができるようにするためである．現在，普及しているIEEE802.11bやIEEE802.11aといった規格は，伝送速度の高速化を目的とした物理レイヤの規定である．

2 IEEE802.11方式無線LANのMACレイヤ技術

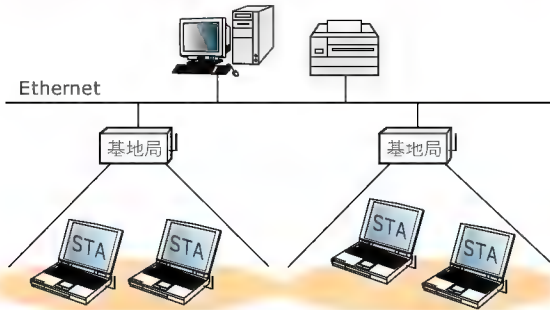
IEEE802.11標準規格においてもっとも基本となるインフラモードのシステム構成は，無線LANアクセスポイント（AP：Access Point）とその電波到達範囲内に存在する無線LAN端末（STA：Station）からなり，これを基本サービスセット（BSS：Basic Service Set）と呼ぶ．構成例を**図7**に示す．また，IEEE 802.11ではアクセスポイントが必要のない無線LAN端末同士で通信を行うアドホックモード（**図8**）もサポートしている．以下では，インフラモードを基本に解説する．

インフラモードでは，各端末とアクセスポイントの間に「帰属」関係が結ばれ，論理的なリンク（回線）が確立される．端末は，唯一帰属先のアクセスポイントを介して，バックボーンネットワークにアクセスし，アクセスポイントは有線-無線間のブリッジング機能を有し，配下の複数の端末を収容する．端末は，特定のアクセスポイントとの無線接続でLAN端末となるが，ユーザーやアプリケーションに対しては，有線でのEthernet接続と同様の環境を提供するため，無線でのアクセスを意識せずに使用できる．また，端末の移動に対しても，各端末が自律的に移

〔図6〕IEEE802.11の検討グループ構成

〔図7〕インフラモードの構成例

• 個々のPC(Station)と基地局による無線LAN

〔図8〕アドホックモード

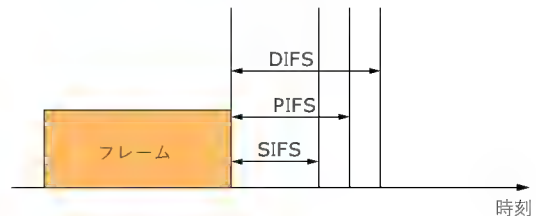
• 端末のみで構成するアドホックネットワーク

動を検出し，異なるBSSのアクセスポイントへ帰属関係を更新する「ハンドオフ」機能を備えており，端末が移動中でも通信を途切れなくすることで，無線環境特有の端末モビリティ(移動性)を確保することができる．

2.1　分散制御による無線チャネルアクセス制御

IEEE802.11方式の基本アクセス手順はDCF(Distributed Coordination Function)と呼ばれており，媒体アクセス方式に前述の自律分散的なアクセス制御ができるCSMA/CA方式が用いられ，アクセスポイント(AP)も端末(STA)も同じ手順でデータ転送を行う．CSMA/CA方式の基本動作は，信号送信を試みようとする無線局がほかの無線局が送信している信号に衝突させないように，事前に無線チャネルの使用状況を確認(キャリアセンス)し，「未使用(アイドル)」であればただちに信号を送信し，「使用中(ビジー)」であればアイドルになるまで送信を延期することである．

チャネルがアイドル状態かどうかを判定するのに，IEEE 802.11ではIFS(Inter Frame Space，フレーム間隔)を規定し，規定された時間以上にわたりチャネルに信号が検出されない場合にアイドルであると判定する．IFSは固定長で規定されているが，長さを複数定義することで，送信局間の優先権を制御している(**図9**)．一方，ビジー状態であった場合には，チャネルが

〔図9〕IFSによる優先制御

• フレーム間隔(IFS: Inter Frame Space)とキャリアセンスで，送信フレームに優先度を与える
• 3種類のフレーム間隔を定義(PIFSはオプション)
  • SIFS：応答などに使用，最高優先
  • PIFS：集中制御に使用
  • DIFS：分散制御に使用，最低優先

アイドルになるまで待ち，その後に衝突回避のためのバックオフアルゴリズムを実行する．

バックオフアルゴリズムとは，複数の無線局が同時に送信を開始してパケットの衝突が起こるのを回避するためのアルゴリズムである．パケット衝突の確率がもっとも高くなるタイミン

コラム1　無線LAN技術関係の参考文献について

無線LANの技術や標準化動向などを解説する書籍は，意外なほど少ない．本章を執筆するにあたり，参考にしたものは，『ワイヤレス・ブロードバンド教科書』(服部 武・藤岡 雅宣編著，IDGジャパン)の第7章，および『最新モバイル・インターネット徹底解剖』(日経コミュニケーション・日経ニューメディア共編，日経BP)の第5章である．いずれも，筆者と同じNTTアクセスサービスシステム研究所の開発者が執筆している．

ほかに多く見られる無線LAN関連の書籍は，現在広く市販されているIEEE802.11bの無線LAN機器の設定法を紹介するハウツーものが多く，技術においてもOFDM方式の変復調技術について紹介するものはあるが，MACレイヤに関して解説するものはほとんどないようである．

この状況は，じつは日本に限ったことではない．洋書でも，IEEE802.11について詳しく解説されているものは，*IEEE802.11 Handbook：A Designer's Companion*(Bob O Hara，Al Patrick著，IEEE Press)と*802.11 Wireless Networks：The Definitive Guide*，(Matthew S. Gast著，O Reilly)程度しか見あたらない．

このようなことからも，無線LANは「活きのいい」技術といえるのかもしれない．もっとも，2002年末～2003年春にかけて，無線LAN関係の書籍が複数出版されるようなので，それらに期待したい．

〔図10〕CSMA/CAによるアクセス制御

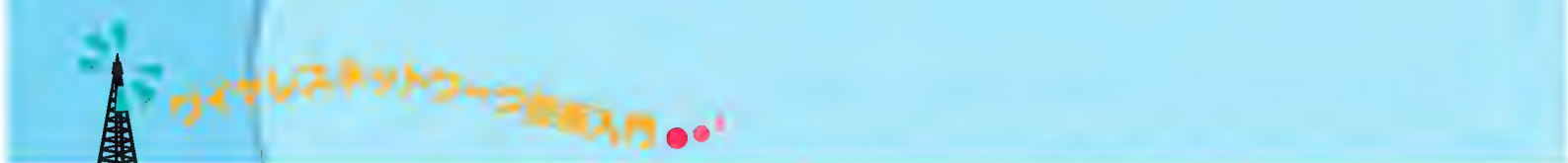
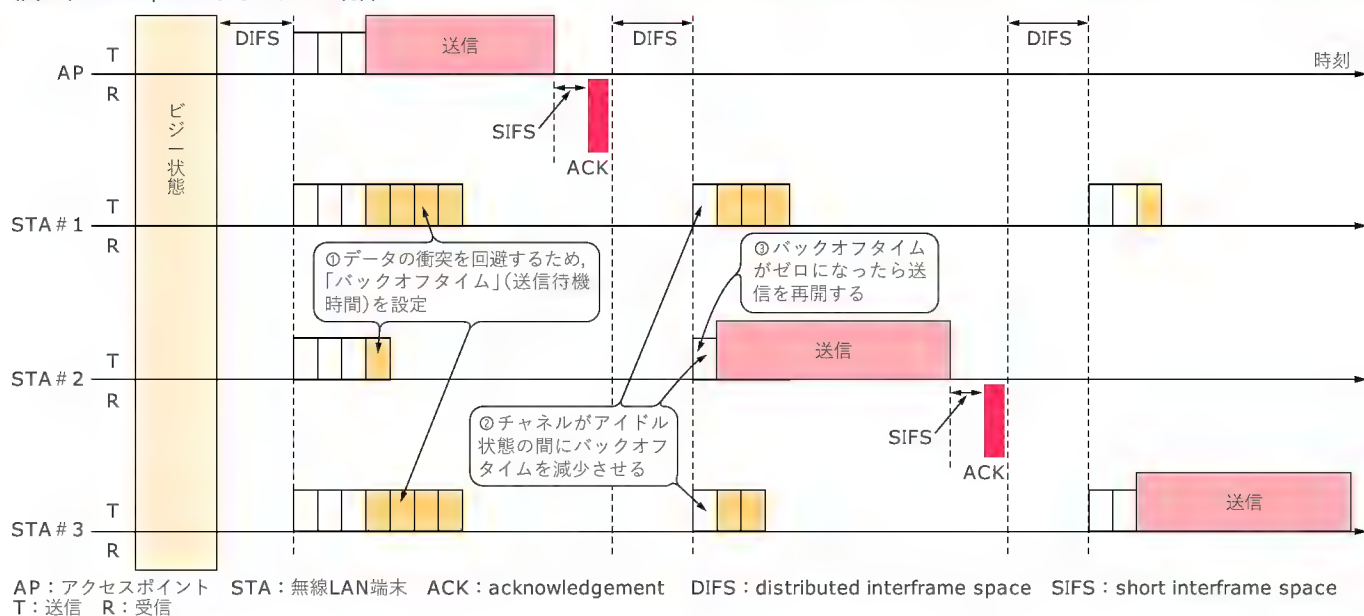
AP：アクセスポイント　STA：無線LAN端末　ACK：acknowledgement　DIFS：distributed interframe space　SIFS：short interframe space
T：送信　R：受信

グは，ある無線局の送信が完了し，チャネルがビジー状態からアイドル状態に変わった直後である．これは，周囲の無線局が送信延期状態であり，アイドル状態になったことを認識した各無線局が一斉に信号の送信を開始することが予想されるからである．

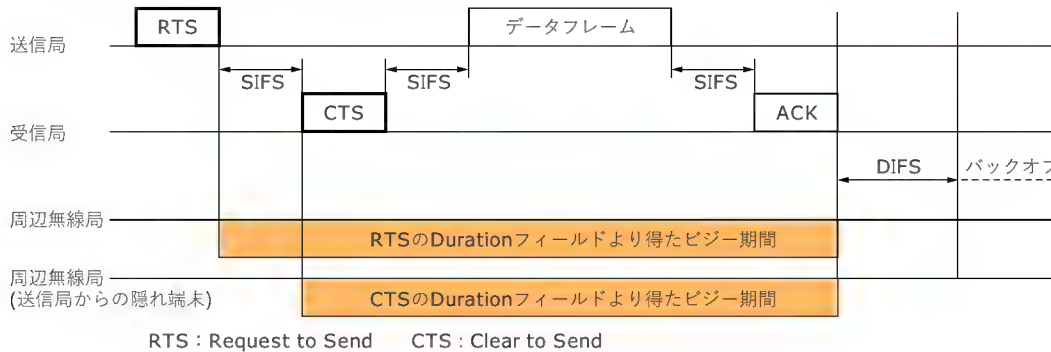
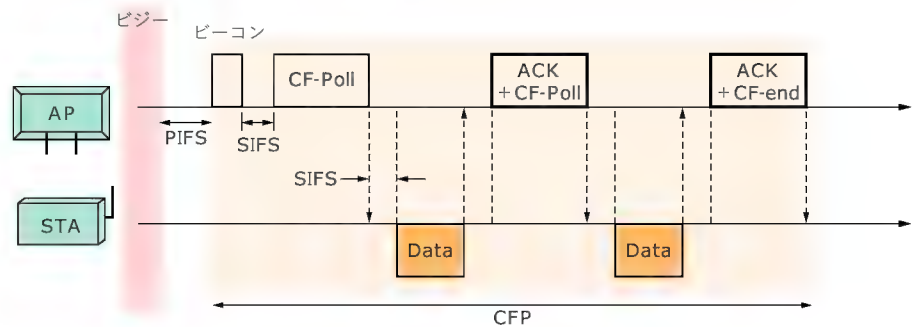
図10を使って，バックオフアルゴリズムを適用したCSMA/CAのアクセス制御法を説明する．APおよびSTAは，常時キャリアセンスを実行し，チャネルの使用状況を監視している．この場合のIFSには，DIFS(Distributed IFS)が用いられる．チャネルがビジー状態であった場合には，ゼロからCW(Contention Window：バックオフアルゴリズムにおいてゼロから一様分布の乱数を発生させる範囲)の範囲内で乱数を発生させて，その値からバックオフタイムを決定する．その後，チャネルがアイドル状態の間にバックオフタイムを減少させ，残りがゼロになった時点で送信を開始する．残り時間がゼロになる前にほかの無線局が送信を開始した場合には，再び送信待機状態となり，チャネルがアイドル状態になった時点から残りのバックオフタイムを再び減少させる．

IEEE802.11のバックオフアルゴリズムは，「2進指数バックオフアルゴリズム」と呼ばれ，発生させる乱数の範囲CWが，最小値CWmin，最大値CWmaxというパラメータで与えられ，初回の乱数発生時にはCWの値はCWminに設定され，再送回数が増えるにつれて2倍に増加し，CWmaxに達した後は一定値となる．トラフィックが多くなり，混みあうにつれてバックオフタイムを増やし，データの衝突を防ぐしくみになっている．

データを正常に受信したAPまたはSTAは，受信完了後にSIFS(Short IFS)間隔後に送信元へACK(Acknowledgement)信号を返す．SIFSはIFSの中でももっとも短く，ACK信号は最優先で送信されていることになる．送信元は，データの送信完了後，規定時間内にACK信号が返ってこない場合には，送信に失敗したと判断し，データを再送する．

CSMAベースのアクセス制御方式には，本質的に避けられない問題として「隠れ端末問題」が存在する．これは，CSMAが互いに無線信号を検知できることを前提とした制御方式であることが要因となっている．たとえば，**図11**のようにA，B，Cの無線局があり，AとBの間に障害物がある(または，距離が離れている)場合を考える．このとき，AとC，BとCは，通信できるが，AとBとは電波が到達せず，通信できない状態にある．この場合，AからCへのデータ送信中に，Aの存在を検知できないBが通信に割り込んでくる(パケットを衝突させてしまう)という問題が生じる．

〔図11〕隠れ端末問題

〔図12〕RTS/CTSによる隠れ端末問題対策

〔図13〕集中制御による無線アクセス制御方法

IEEE802.11では，この問題を避けるために「RTS(Request To Send)/CTS(Clear To Send)手順」を用意している(**図12**)．これは，データ送信側が事前にRTS信号を送信し，これを受けた受信側がCTS信号を送信することで，周囲のAPおよびSTAにチャネルの使用を予告し(仮想キャリアセンス)，パケットの衝突を回避するものである．

2.2 集中制御

IEEE802.11のアクセス制御方式には，前述のDCFのほかに，PCF(Point Coordination Function)方式がオプションとして規定されている．PCFは，APがSTAの送信を制御するため，通信品質の制御が必要な場合に適している．

PCFでは，APが各局の送信権を集中的に制御する．APは周期的にCFP(Contention Free Period)と呼ばれる非競合アクセス期間を設定し，この期間における自律分散的なチャネルアクセスを規制する．APはCFP期間中に，事前に登録したSTAに対してポーリングを行うことでSTAへ送信を許可する(**図13**)．APからのポールフレームに続くSTAのデータ送信は，SIFS間隔で行われる．ポールフレームをAPが特定のSTAに送ることで，そのSTAは他のSTAよりも優先度の高い送信権を得ることができ，ほかのデータフレームと競合することなく，安定したデータ送信が可能となる．

近年，無線LANを利用した動画像などのストリーミング伝送などへの要求が高まり，後述するタスクグループe(TGe)では，このポーリングモードを改良したMACレイヤプロトコルが検討されている．

2.3 マネジメント

IEEE802.11のMACレイヤでは，前述のアクセス方式に加え，マネジメント機能が規定されている．基本的には，STAが特定のAPを介してLANに収容されるまでの一連の動作に関する管理手順となっている．各BSSのAPは，自局の所属するBSSの運用情報を「ビーコン信号(無線標識信号)」と呼ばれる制御用パケット信号によって，周期的に報知する．STAは，すべての無線チャネル上に送出されているビーコン信号を受信し，無線通信の品質が安定していることと，収容されるべきLANに接続されていることを確認したうえで最適なAPを選択する．続いてSTAは，選択したAPに対して，認証を要求し，APは自局を介してネットワークへ接続を希望するSTAとして正当であるかどうか確認する．最後に，AP-STA間の帰属関係を結ぶため，互いに管理用信号をやり取りし，APがそのSTAを管理テーブルに登録することで，LAN端末として正式に扱われる．

2.4 セキュリティ

セキュリティ機能には，MACレイヤでの認証機能と暗号化ならびに改ざん防止機能が規定されている．

認証方式には，オープンシステム方式と事前共有鍵方式が規定されている．オープンシステム方式では，認証フレームを相互にやり取りするのみであるが，事前共有鍵方式ではチャレン

ジレスポンス(チャレンジコードと呼ばれる1回かぎりの数字をもとにパスワードを暗号化してやり取りする)による認証を行い，セキュリティを高めている．

一方，暗号化方式として，WEP(Wired Equivalent Privacy)が規定されている．WEPは，米国RSAセキュリティが開発した共通鍵暗号のRC4を使用した方式である．たがいに共通の鍵を使って相手を認証するとともに，データを暗号化する．

具体的には，24ビットの初期ベクトルと40ビットの秘密鍵から暗号化のための鍵を生成し，RC4のアルゴリズムにしたがい暗号化する．また，暗号化したフレームには改ざん防止用のフィールドを設定する．元のデータからCRC-32により算出したICV(Integrity Check Value)を設定し，受信側でチェックすることで，途中でデータが改ざんされたかどうかを判断できる(**図14**)．

近年，WEPはその脆弱性が指摘され，より強いセキュリティを望む声が高い．後述するタスクグループi(TGi)では，この脆弱性の改善に関して検討が進められている．

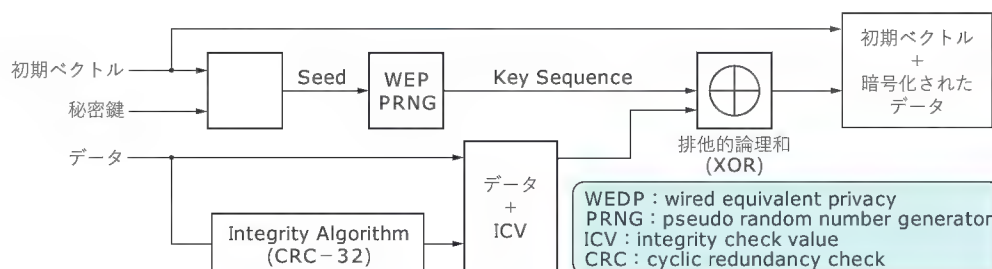
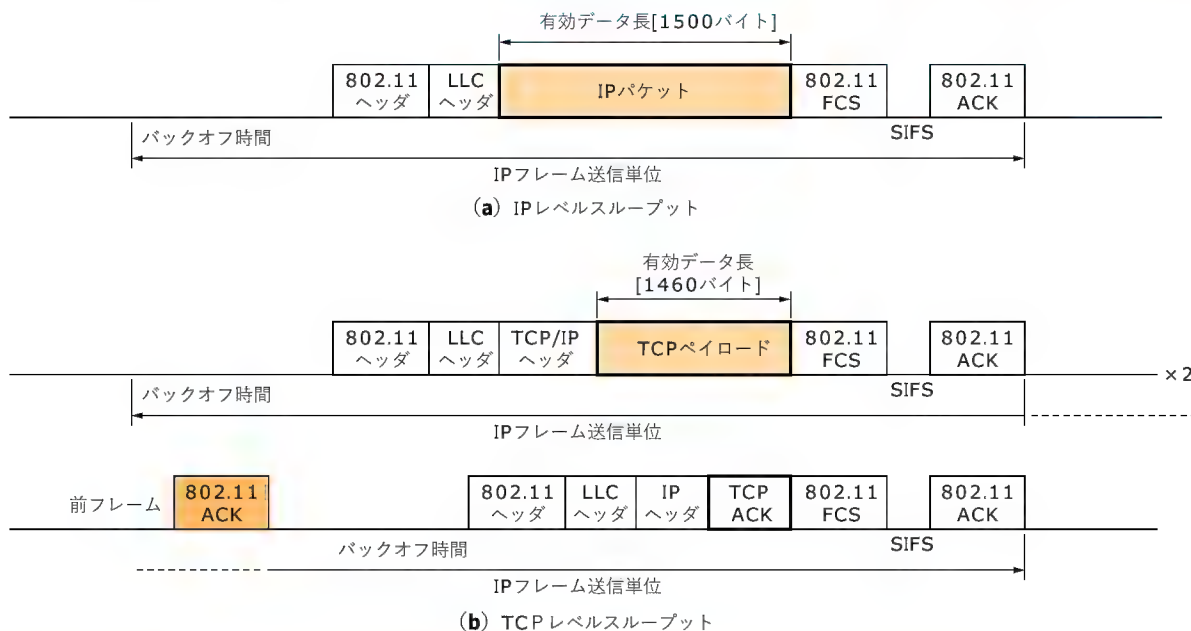
2.5 スループット特性

一般に無線LAN機器の伝送速度は，物理レイヤの伝送速度のことを呼ぶ場合が多い．たとえば，後述のIEEE802.11bでは，最高11Mbpsと，カタログなどには表示されている．しかし，実際にデータを伝送するにあたって，**図15**に示すような各パケットに付加されるプリアンブルやヘッダ，CSMA/CA制御のためのIFSやバックオフタイム，信号到達確認のためのACK信号など，データ伝送のためのオーバヘッドとなっている部分があり，IPレベル・TCPレベルでのスループットは物理レイヤの伝送速度より低い値となる．

IEEE802.11aとIEEE802.11bについて，スループットの理論的な値を算出したものを**表2**に示す．このように，IEEE802.11bの11Mbpsモードと IEEE802.11aの6Mbpsモードは，スループットという観点からはあまり差がない．また，IEEE802.11aの24Mbpsモードや36Mbpsモードと比較すると，3～4倍の差となっている．これは，IEEE802.11bのプリアンブル部およびヘッダ部が，IEEE802.11aのそれに比べ長いということに起因する．

3 IEEE802.11方式無線LANの物理レイヤ技術

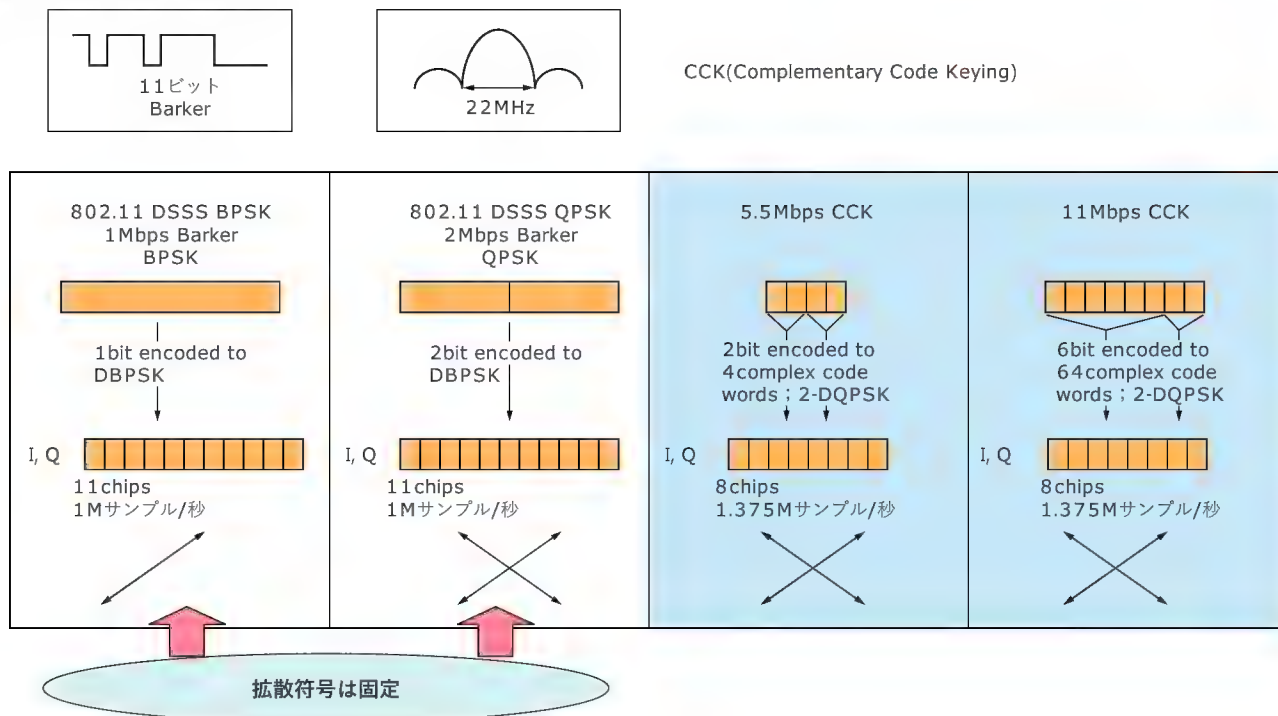
IEEE802.11の物理レイヤは，PMD(Physical Medium Dependent)副層とPLCP(Physical Layer Convergence Protocol)副層に分かれている．PMD副層では，変復調方式関連を規定しており，前述の① DS-SS方式，② FH-SS方式，③赤外線方式が

〔図14〕WEPによるデータの暗号化

〔図15〕上位レイヤでのスループット計算

〔表2〕最大スループット

Data Rate [Mbps]	IEEE802.11a								IEEE802.11b Long PLCP format			
	6	9	12	18	24	36	48	54	1	2	5.5	11
IP フレーム	5.37	7.78	10.0	14.1	17.6	23.7	28.5	30.8	0.91	1.73	3.99	6.37
転送速度効率	0.90	0.86	0.84	0.78	0.73	0.66	0.59	0.57	0.91	0.86	0.72	0.58
TCP フレーム	4.94	7.05	8.97	12.3	15.1	19.7	23.0	24.5	0.85	1.56	3.40	5.11
転送速度効率	0.82	0.78	0.75	0.69	0.63	0.55	0.48	0.45	0.85	0.78	0.62	0.46

約3～4倍

〔図16〕DS-SS方式とCCK方式

CCK(Complementary Code Keying)

規定されている．PLCP副層はPMD副層とMACレイヤの間に位置し，PMD副層で規定されている3種類の変復調方式の違いを吸収することで，MACレイヤと物理レイヤ間のインターフェースを統一する役目を果たす．

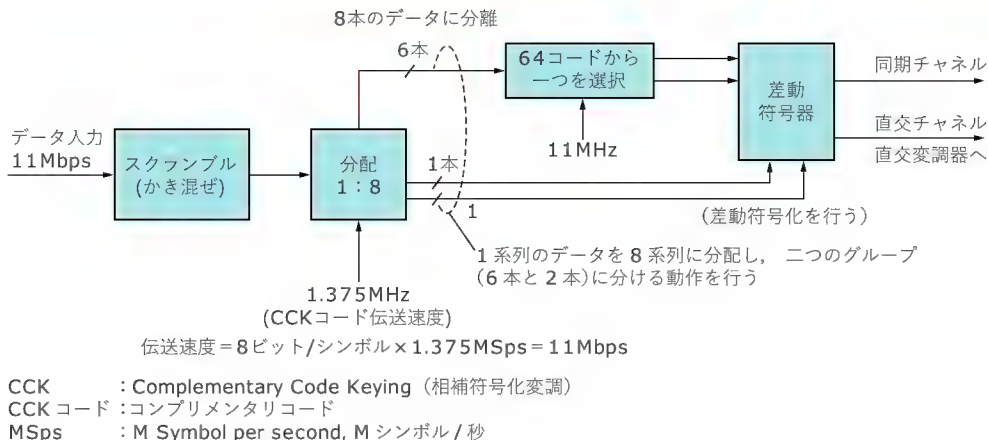
3.1 IEEE802.11b

1997年にオリジナル規格IEEE802.11が完成した後，IEEE802.11ワーキンググループにおいて物理レイヤの高速化に関する検討が始まった．このとき，2.4GHz帯の周波数を使用して物理レイヤの高速化を検討したのが，タスクグループb(TGb)であり，1999年にインターシル(Intersil)社とルーセント・テクノロジ(Lucent Technologies)社の共同提案によるCCK(Complementary Code Keying，相補符号変調)方式を採用し，最高11Mbpsの伝送速度を実現するIEEE802.11b規格が完成した．現在もっとも普及している無線LAN機器の規格であり，現在オフィスや家庭で使用されているもののほとんどが，このIEEE802.11b規格に準拠したものである．

IEEE802.11b規格は，従来のIEEE802.11規格におけるDS-SS方式を拡張した位置付けであり，後方互換性を保ち，伝送速度はDS-SS方式の1Mbpsと2Mbpsに加えて，5.5Mbpsと11Mbpsが規定されている．DS-SS方式は，スペクトル拡散変調に11ビットのバーカー(Barker)符号を用いているが，IEEE802.11b規格では，コンプリメンタリコードを使ったCCK方式を追加し，高速化を実現している(**図16**)．

CCK変調器は，**図17**のような構成となっている．11Mbpsのデータ入力は，スクランブルされた後，1.375MHzのクロック信号にしたがって8本の信号に分配される．そのうち6本の信号は，64種類のCCKコードから特定の一つのコードを選択することで6ビットの情報を伝送する．また，同時に分配部での残り2本の系列は差動符号器に入力され，コード全体に位相回転を与えることになる．これにより2ビットの情報を伝送する．

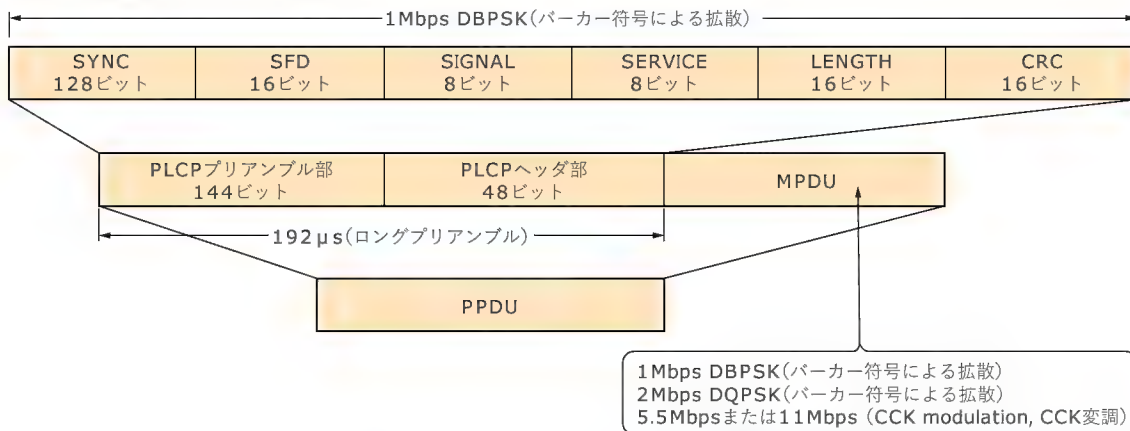
したがって，1.375MHzで6＋2＝8ビットの情報をもつ1シンボルを伝送することから，情報伝送速度が11Mbpsで無線区間に送出されることになる．5.5Mbpsモードでは，分配部で4分配し，2本の信号から4種類のCCKコードを選択することとす

〔図17〕CCK方式の変調器構成

る．そのため，1.375MHzで2＋2＝4ビットの情報を伝送することから，5.5Mbpsの伝送速度となる．

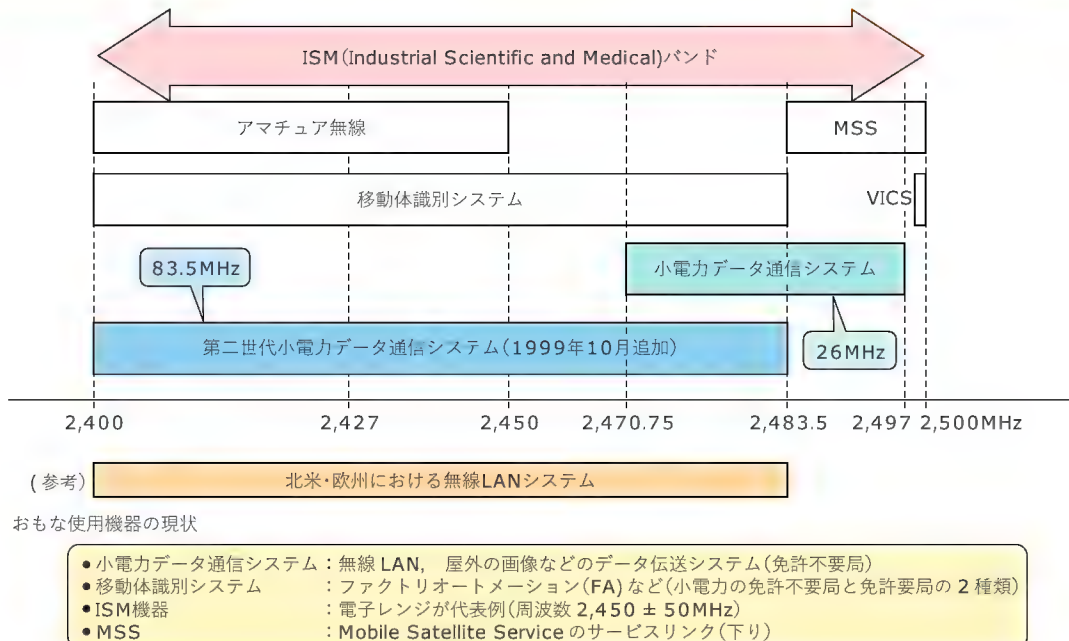
IEEE802.11bでは，前述のように4種類の伝送速度が規定されているが，どの伝送速度モードを使用するかは送受信局間の伝送路の状況に応じて動的に決まる．この機能が，レートアダプテーションである．詳細は標準規格の規定対象外とされてはいるが，一般に過去の送信結果の履歴から伝送路の品質を判断し，伝送速度を決定する方法が用いられている．

レートアダプテーション機能により，送受信局間の伝送路の品質が良い場合には，高速伝送が可能となる．逆に伝送路状態が悪い場合には，外乱に強い低い伝送速度を使用し送受信局間のコネクティビティを確保する．

レートアダプテーション機能を用いる場合，伝送速度がパケットごとに異なる可能性がある．このため，受信局側で復調する際に，送信側で用いられた伝送速度(変調方式)の情報が必要となる．IEEE802.11bでは，プリアンブル部，ヘッダ部の伝送

〔図18〕IEEE802.11b無線LANのパケット構成

※Barker code：バーカー符号．自己相関特性が優れた11ビットの符号長からなる拡散符号

略　語	フルスペル	説　明
SYNC	Synchronization	同期処理信号
SFD	Start Frame Delimiter	フレーム開始，物理層に依存した有効フレームの先頭を示す
SIGNAL	―	データ部の伝送速度を示すフィールド
SERVICE	―	高速変調方式を識別するフィールド（CCK，PBCC）
LENGTH	―	データ部を送信するための時間（ms単位）
CRC	Cyclic Redundancy Check	巡回冗長検査，誤り訂正方式の一つ
PLCP	Physical Layer Convergence Protocol	物理層コンバージェンスプロトコル
MPDU	MAC Protocol Data Unit	MACプロトコルデータ単位
PPDU	PLCP Protocol Data Unit	PLCPプロトコルデータ単位
DBPSK	Differential Binary Phase Shift Keying	差動2相位相偏移変調
DQPSK	Differential Quadrature Phase Shift Keying	差動4相位相偏移変調
CCK	Complementary Code Keying	相補符号変調

〔図19〕日本の2.4GHz帯周波数割り当ての状況

速度を1Mbpsに固定し，ヘッダ領域内にデータ部分で使用する伝送速度の情報を設定することで，受信側でデータを正しく復調できるようにしている(**図18**)．また，旧来のDS-SS方式のみサポートしている無線局に対する後方互換性もこれにより確保されている．

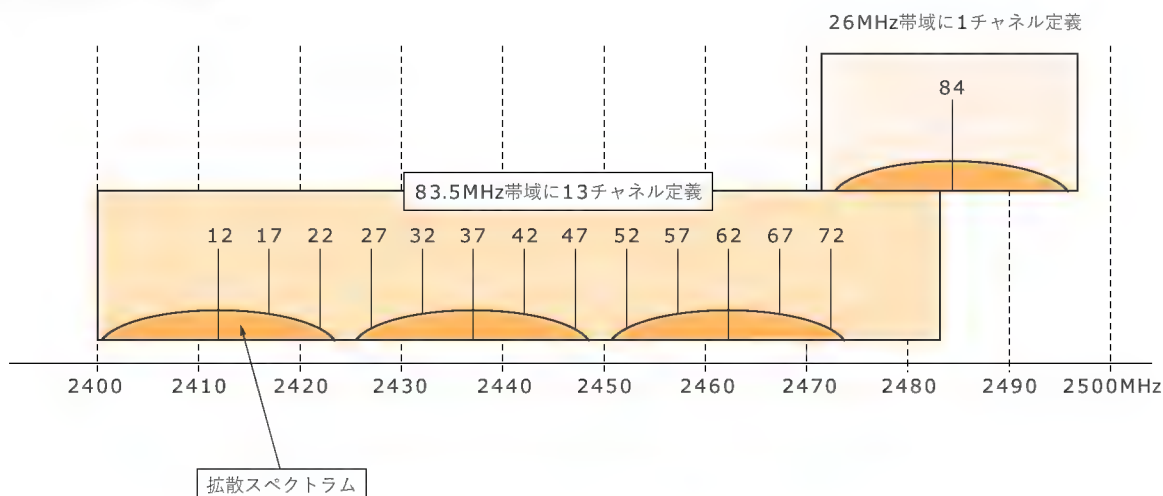
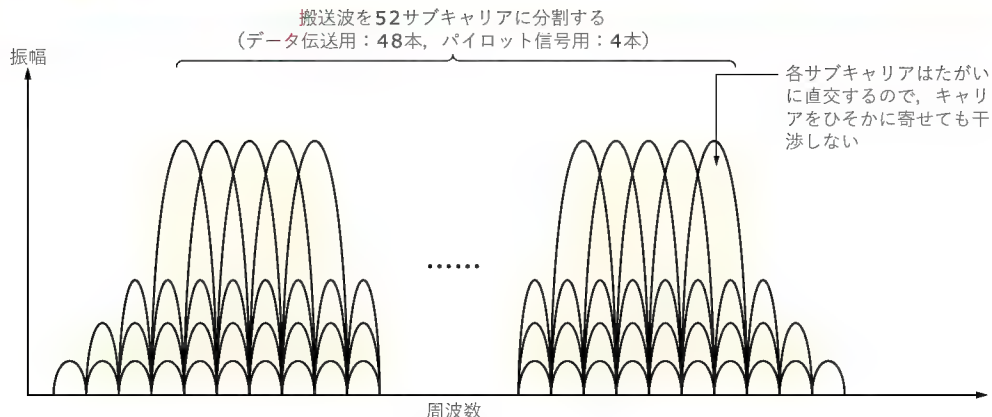
● 2.4GHz帯周波数割り当ての状況

日本における周波数割り当ての状況を**図19**に示す．2.4GHz帯は前述のようにISMバンドと呼ばれており，名前のとおり産業用電子加熱装置，電子レンジ，温熱治療器などが通信以外の目的で用いられている．また，通信の分野でも，アマチュア無線，移動体識別システム(電子タグ)，Bluetoothなどが同じ周波数帯を共用している．

日本では，当初26MHzの帯域が無線LAN用として開放され，その後1999年10月に省令改正がなされ，日米欧共通バンドである83.5MHzの帯域が無線LANで利用可能となった．この帯域拡大とIEEE802.11bの標準化のタイミングが一致し，急速に市場を拡大していった．

さらに，後述するIEEE802.11g規格を日本でも利用可能とするため，2002年2月にこの83.5MHzの帯域について，OFDM変調方式の適用を認める省令改正がなされた．

図20にDS-SS方式の場合のチャネル配置例を示す．IEEE 802.11規格では，5MHz間隔でのチャネルが規定されているが，ISMバンドでの電波干渉を避ける目的で中心周波数をずらせるようにしたものである．したがって，他システムからの干渉がない場合には，このように83.5MHzの中に3波配置するのが標準的である．

〔図20〕DS-SS方式のチャネル配置

〔図21〕OFDMの周波数スペクトル

3.2 IEEE802.11a

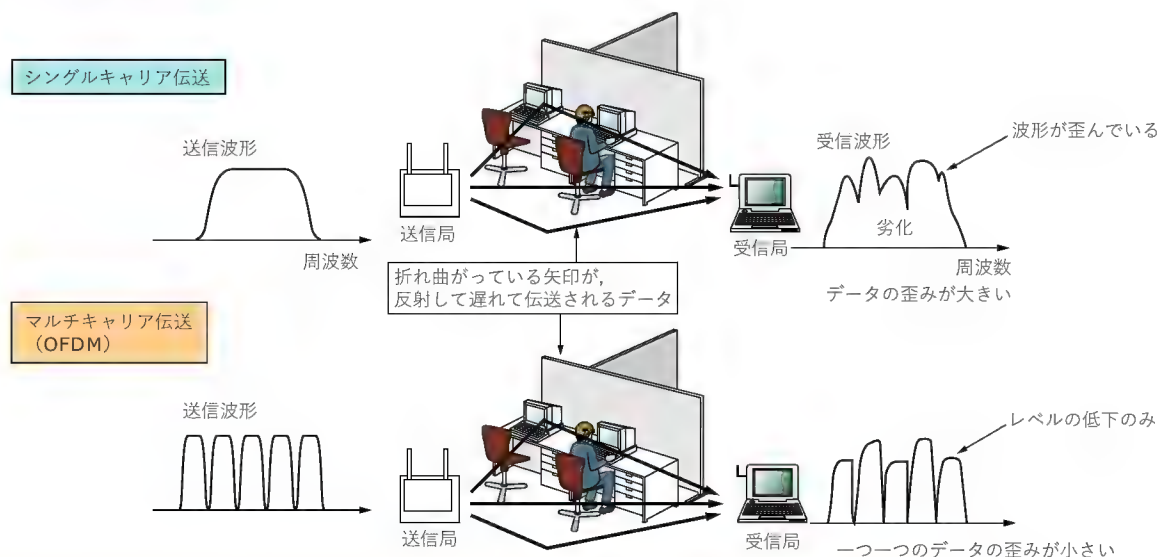
1997年に米国FCC（Federal Communications Commission，アメリカ連邦通信委員会）が，U-NII（Unlicensed National Information Infrastructure）帯と称する5.15～5.35GHz，5.725～5.825GHzの合計300MHzの帯域を免許不要な無線LAN・無線アクセス用に開放したことを受けて，TGbにおける2.4GHz帯の高速化と並行する形で5GHz帯を用いた20Mbps以上の伝送速度を実現する物理レイヤの検討が始められた．この検討を行ったのが，タスクグループa（TGa）であり，1999年にIEEE 802.11a規格を完成した．

IEEE802.11aもIEEE802.11bと同様に物理レイヤの規格であり，MACレイヤは，従来のIEEE802.11を使用する．近年，多数のメーカーが，IEEE802.11a準拠製品の出荷を始めており，次世代を担う方式として注目を集めている．

IEEE802.11aでは，変調方式としてNTTとルーセント・テクノロジの共同提案によるOFDM(Orthogonal Frequency Division Multiplexing，直交周波数分割多重)方式を採用している．OFDM方式は，欧州でDAB（Digital Audio Broadcasting，ディジタルオーディオ放送）規格やDVB（Digital Video Broadcasting，ディジタル映像放送）規格として標準化されているが，IEEE802.11aでは，パケットモードに適用した点に特徴がある．

OFDM方式はマルチキャリア伝送方式の一種であり，複数のサブキャリアを周波数軸上に直交条件を満たすように密に配置する（**図21**）．このサブキャリアを配置・合波するアルゴリズムには，IFFT（Inverse Fast Fourier Transform，逆高速フーリエ変換）が用いられる．

OFDM方式の特徴として，サブキャリアを密に配置できるため，高い周波数利用効率が得られることがあげられる．また，高速信号の伝送時に問題となるマルチパス伝搬による周波数選択性フェージングが原因となる波形の歪みに対して強い耐性をもつ点も，大きなメリットとなる．従来一般に用いられるシングルキャリア伝送方式では，波形の歪みを補正するために等化器（イコライザ）を使用する必要があり，回路が複雑かつ大規模になるという問題がある．一方，マルチキャリア伝送方式であるOFDMでは，波形の歪みを部分的なサブキャリアのレベル低下という形にみなせる．レベル低下，すなわちS/N比の劣化は，

〔図22〕マルチパス環境に強いOFDM方式

〔図23〕多重干渉波の影響を軽減するガードインターバル

最適FFT位置について
無線LANの送信側で送られた信号は，受信側無線LANに直接波として届くものと，多重反射して遅延波として届くものの2種類が合成された形になる．とくに遅延波は1か所に反射するだけでなく，多くの壁や天井に反射して届くため，多くの遅延波で構成されてる．受信側で最適受信するためには，この直接波と遅延波が混在する環境下で，いかにタイミングよく高速フーリエ変換（FFT）の回路を動作させるかが重要である．そこで，もっともタイミングの良いFFT位置とされるのが，最適FFT位置である

事前に伝送データに誤り訂正符号化とインタリーブを施すことで補償する．このように，周波数選択性フェージングによる波形の歪みを比較的簡単な方法で補正することができる（**図22**）．

また，OFDM方式の特徴として，多重波による符号間干渉の影響を軽減するため，各シンボルにガードインターバル（GI）と呼ばれる冗長信号が挿入される．高速伝送信号を単に複数のサブキャリアに分割するマルチキャリア伝送方式だけにOFDM方式の特徴があるのではなく，このガードインターバルが多重波干渉の軽減という観点から非常に重要である．

多重波は，直接波以外に多くの遅延波（周辺の物体に反射して届く電波）を含んでいる．受信側で，各サブキャリア信号を分波する場合に，単純にFFT（Fast Fourier Transform，高速フーリエ変換）してしまっては遅延波が含まれるため，きちんと分波できない．これを解決するため，**図23**中に「コピー」とあるように，各OFDMシンボルの後端部分を先端部分にコピーし，信号が周期的に連続となるように巡回的な構成となるようにしたものが，ガードインターバルである．これにより，受信側でFFT処理を行う場合にマルチパスによる遅延波が信号に含まれていても，ガードインターバルの範囲内であれば，正常にFFTによる分波が可能となる．IEEE802.11aでは，ガードインターバルを800nsとし，二乗平均値で100～200nsの遅延分散をもつ大規模なショッピングモールや工場などの環境下にも適用できる設定となっている．

IEEE802.11aでは，各サブキャリアの変調方式として，BPSK，QPSK，16QAM，64QAMが規定されている．また，誤り訂正の符号化率として1/2，2/3，3/4が規定されており，これらの組み合わせによって，6～54Mbpsの8種類の伝送速度が規定されている（**表3**）．なお，IEEE802.11aにおいてもレートアダプテーション機能により，送受信局間の伝送路状況に応じて，最適な伝送速度を選択して使用する．

IEEE802.11aのパケットフレーム構成を**図24**に示す．ここでプリアンブル部は，FFTのシンボル同期などの各種同期を行うため固定パターンとしている．またヘッダ部は，信頼性の向上とすべての無線局に共通に受信できるようにするため，最低伝送レートの6Mbpsモードで伝送している．

● 5GHz帯周波数割り当ての状況

5GHz帯における周波数割り当ての状況を**図25**に示す．日本では，2000年3月に省令改正が行われ，5.15～5.25GHzの100MHzを屋内限定という条件で，無線LAN，無線アクセスなどに開放した．「屋内限定」の理由は，同じ周波数帯を移動体衛星

〔表3〕IEEE802.11aのシステムパラメータ

変調方式	OFDM方式 各サブキャリアの変調方式 BPSK，QPSK，16QAM，64QAM
サブキャリア数	52サブキャリア(4パイロット信号を含む) 64ポイントFFT
誤り訂正方式	拘束長 $K=7$，符号化率 $R=1/2$，2/3，3/4の畳み込み符号化・ビタビ復号方式 シンボル内インタリーブ
伝送レート	6Mbps (BPSK，$R=1/2$) 必須 9Mbps (BPSK，$R=3/4$) オプション 12Mbps (QPSK，$R=1/2$) 必須 18Mbps (QPSK，$R=3/4$) オプション 24Mbps (16QAM，$R=1/2$) 必須 36Mbps (16QAM，$R=3/4$) オプション 48Mbps (64QAM，$R=2/3$) オプション 54Mbps (64QAM，$R=3/4$) オプション
チャネル配置	4(100MHz)，8+4(200+100MHz：米国) 20MHzチャネル間隔

通信システムと共用しているため，国際的に電波干渉を避ける審議がなされ，国際勧告化されたことによる．

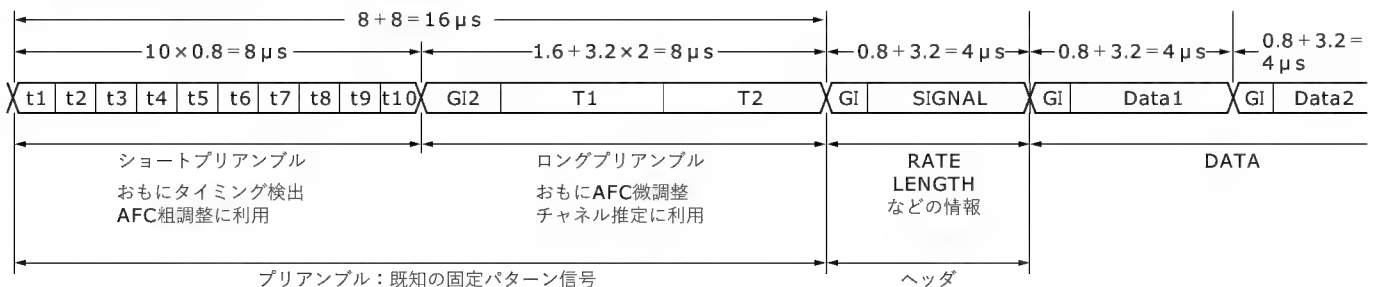
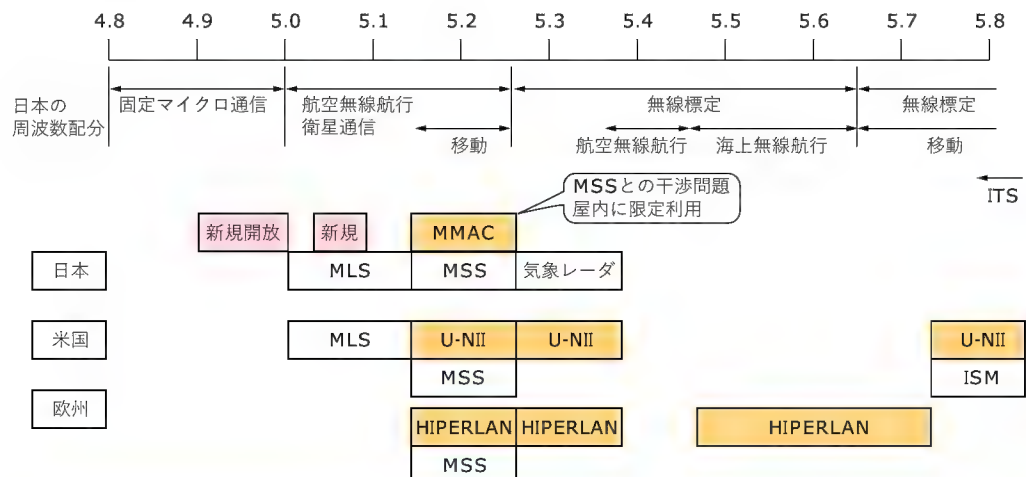
日本および米国における5GHz帯無線LANのチャネル配置を**図26**，**図27**に示す．米国の場合，5.15～5.35GHzの200MHz帯が連続しており，5.25GHzから上の100MHzでは屋外でも条件付きで使用可能となっている．この200MHz帯域に対して，帯域外の不要輻射の制限から，日米のチャネル配置には10MHzのずれが生じている．

欧米では屋外で使用可能な帯域があることもあり，日本においても屋外で使用可能な周波数を望む声が高まっていた．情報通信審議会では，2001年10月の諮問より5GHz近傍の周波数利用の審議がなされてきた．その結果，2002年9月に省令改正され，屋外でも利用可能な固定無線アクセスシステムおよび無線LANに代表される移動無線アクセスシステムを対象として4.9～5.0GHz，5.03～5.091GHzの二つの周波数帯が開放された．

しかし，新たに開放された周波数帯では，通信事業者によるライセンスバンド(電波免許が必要な帯域)となり，公衆無線LANサービスや固定無線アクセスとして利用するものとなっている．

IEEE802.11の標準化動向

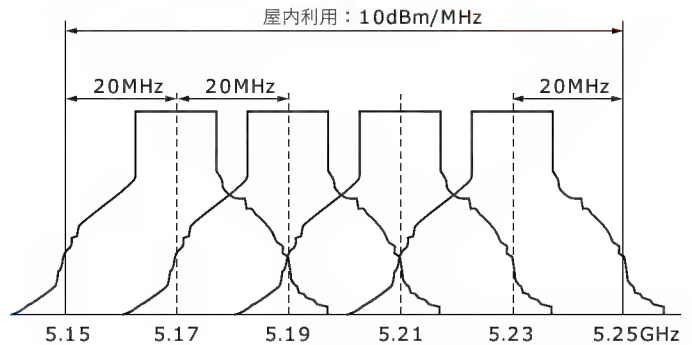
IEEE802.11a/bの高速無線LAN標準規格が承認された後も，

〔図24〕IEEE802.11aのパケットフレーム構成

〔図25〕5GHz帯周波数割り当ての状況

IEEE802.11ワーキンググループでは，物理レイヤのさらなる高速化とMACレイヤの高機能・高効率化に向けた検討が進められている．

ここでは，2002年10月末時点での状況をまとめた．最新の情報についてはIEEE802.11ワーキンググループのWebページ(http://grouper.ieee.org/groups/802/11/index.html)を参照してほしい．

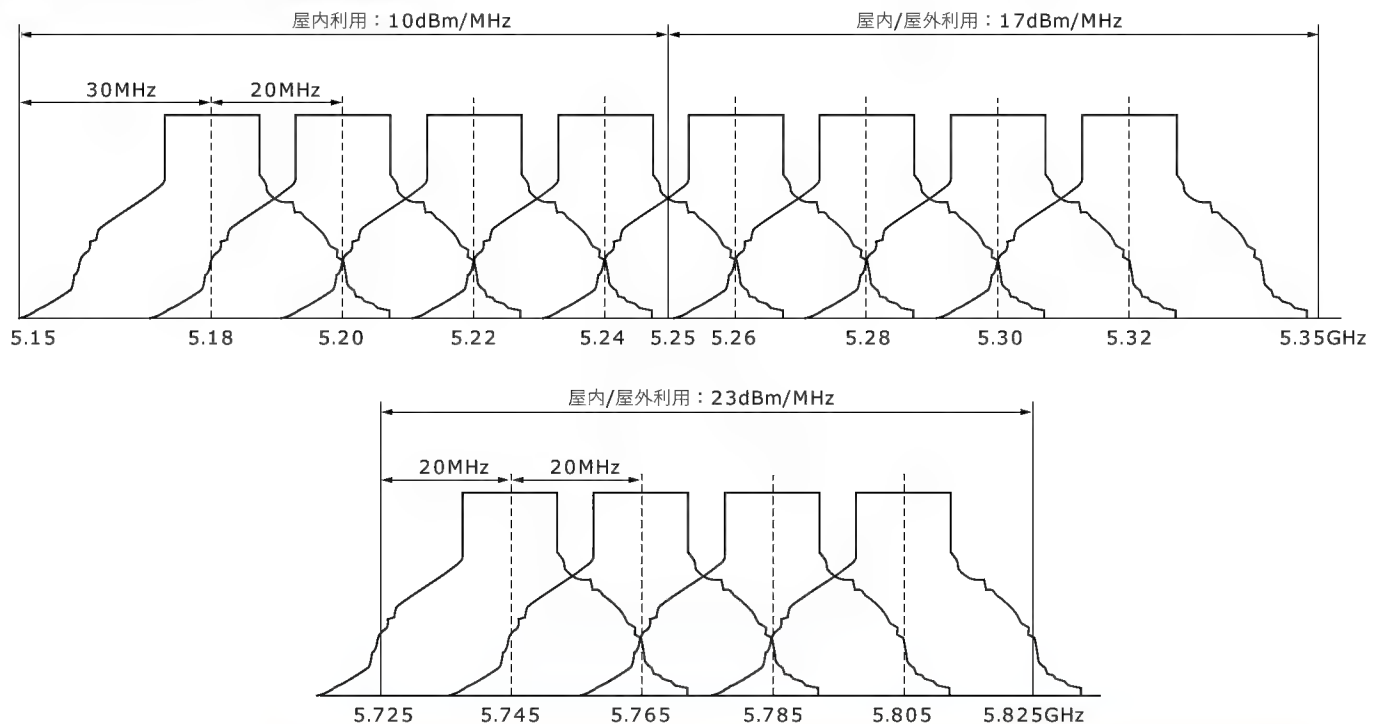
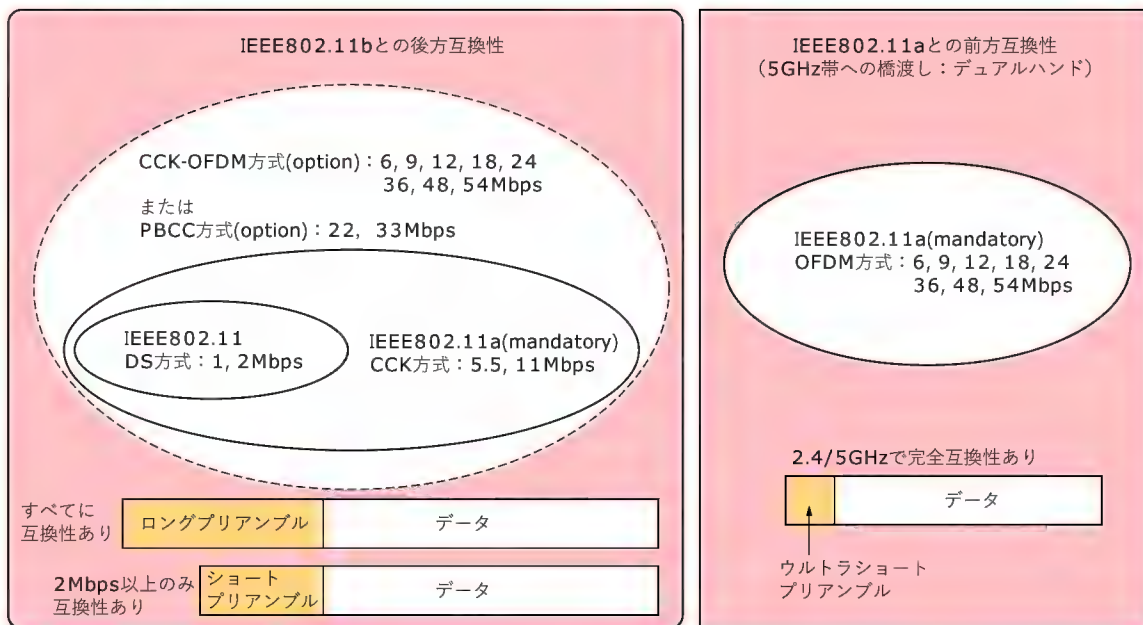
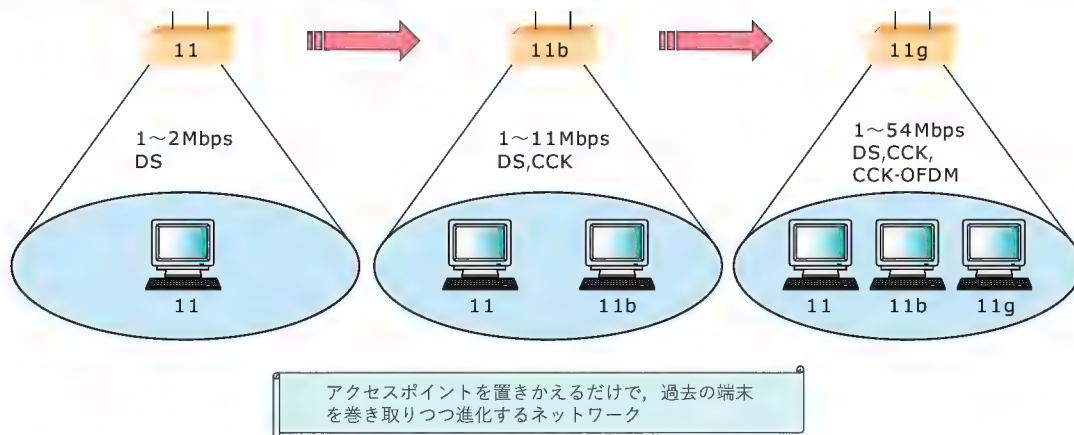
4.1 タスクグループg(TGg)：IEEE802.11bの高速化

TGgは，IEEE802.11b規格との後方互換性を保ちつつ，20Mbps以上の伝送速度を提供する物理レイヤの実現を目的として検討が進められてきた．ここまで，米国インターシル社の提案するCCK-OFDM方式と，米国テキサス・インスツルメンツ(Texas Instruments)社の提案するPBCC(Packet Binary Convolutional Coding)方式との間で激しい議論がなされていたが，既存方式との互換性などを考慮した結果，IEEE802.11aで採用されたOFDM方式と，IEEE802.11bで採用されたCCK方式の両方を必須方式とし，前述のCCK-OFDM方式とPBCC方式はオプションとすることでまとまっている．

しかしながら，異なる変調方式を使用する局間でキャリアセンスが正常に機能しない可能性があるため，パケット衝突によるスループットの低下をまねく恐れがある．そこで防止策として，MACレイヤで規定されているRTS/CTS手順などを用いた衝突回避手段を盛り込むこととしている．現在，TGgにて作成されたドラフトが，ワーキンググループにて承認された状態であり，今後，標準規格として承認されるべき手続きをとるため，上部組織に送られる予定である．

〔図26〕IEEE802.11aのチャネル配置(日本)

ここでCCK-OFDM方式とは，IEEE802.11b規格標準のCCK方式に加えて，IEEE802.11a規格標準のOFDM方式を加える形で拡張した方式である．IEEE802.11bとの互換性を保つため，ヘッダ部までをCCK方式とし，データ部を高速かつ安定な伝送品質を実現するOFDM方式としたもので，まさに両者の融合方式となっている．

一方，PBCC方式は，IEEE802.11b規格においてオプション規定されている方式で，CCK方式同様11Mbpsが定められているが，ここでは，22Mbps，33Mbpsに拡張されている．PBCC方式は，基本的には畳み込み符号器とスクランブラを組み合わせたシングルキャリア方式で，等化器使用を前提とした方式で

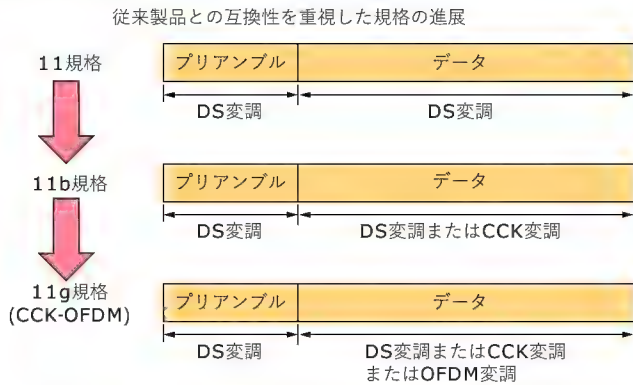
〔図27〕IEEE802.11aのチャネル配置(米国)

〔図28〕IEEE802.11gの役割

〔図29〕2.4GHz帯IEEE802.11b/gのネットワーク進化

コラム2 講演や展示会でのQ&Aから

セミナなどの講演や展示会などの説明の際に受ける質問でもっともFAQといえるものは，「どれくらい電波(信号)は届くのか？」であろう．この質問に正確に答えるのは困難である．それは，無線LAN機器の設置環境や機器の仕様(アンテナや送信出力)などによって答が大きく変わってしまうからである．あたりさわりのない答として「環境によって異なりますが，オフィスのような環境で50m前後，木造の一戸建てなら一台のアクセスポイントで一軒がカバーできると思います」ということになる．しかし，本当のところは設置してみないとよくわからないのである．

また，「隣の部屋で同じチャネルを使うと混信(干渉)しないのか？」という質問も多い．これは，本文中で説明したCSMA/CAのしくみをご理解いただければ，たがいに無線チャネルのリソースを分け合うことで「影響」はあるものの，システムとして使用することはできるということがわかると思う．しかし，産業用機器や医療用機器などからの干渉を受けて，スループットの低下や最悪時には通信途絶といった事態が起こることがあるのも事実である．

「無線LANの電波は身体に影響あるか？」といった質問も，時折受ける．これは，電車などで携帯電話やPHSの使用自粛や医療機器などへの影響を心配してのことと思われる．無線LAN機器は，一般的な製品で送信出力が10mW～50mW程度であり，PHSの80mW，携帯電話の数100mWと比べて非常に低い電力輻射であることから，まず常識的には問題ないと考えられる．また，人体(とくに頭部)に近接して使用しないという状況からも，影響が少ないといえるであろう．

〔図30〕IEEE802.11b/gの進化

〔図31〕IEEE802.11b/gの進化における課題

ある．

IEEE802.11gには，**図28**に示すように二つの役割がある．

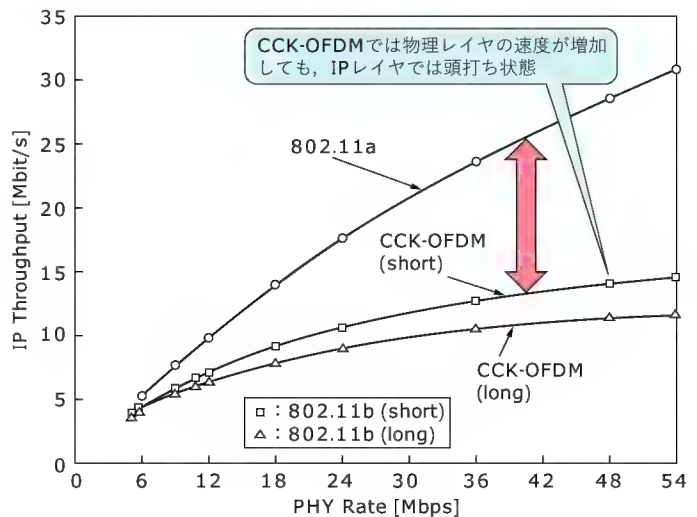
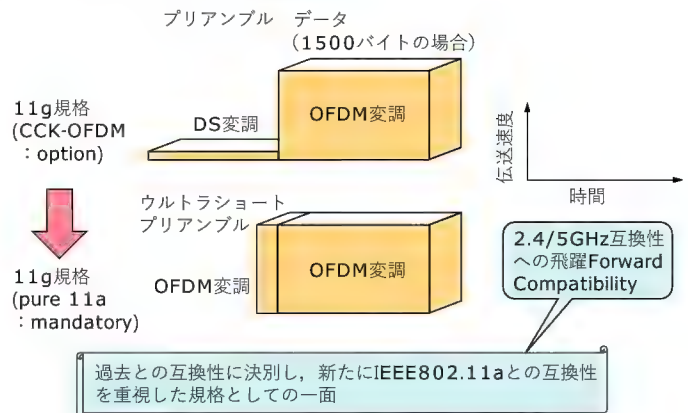
① 2.4GHz帯における従来システムであるIEEE802.11およびIEEE802.11b方式との後方互換性をもちながら，さらなる高速化をはかる

② 5GHz帯システムであるIEEE802.11a方式との将来の互換性(前方互換性)をはかる

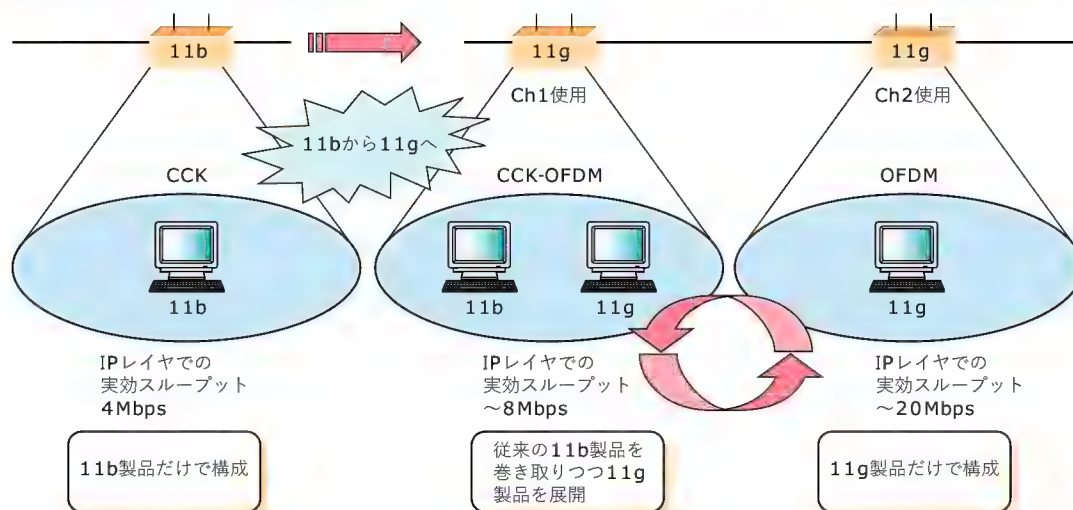
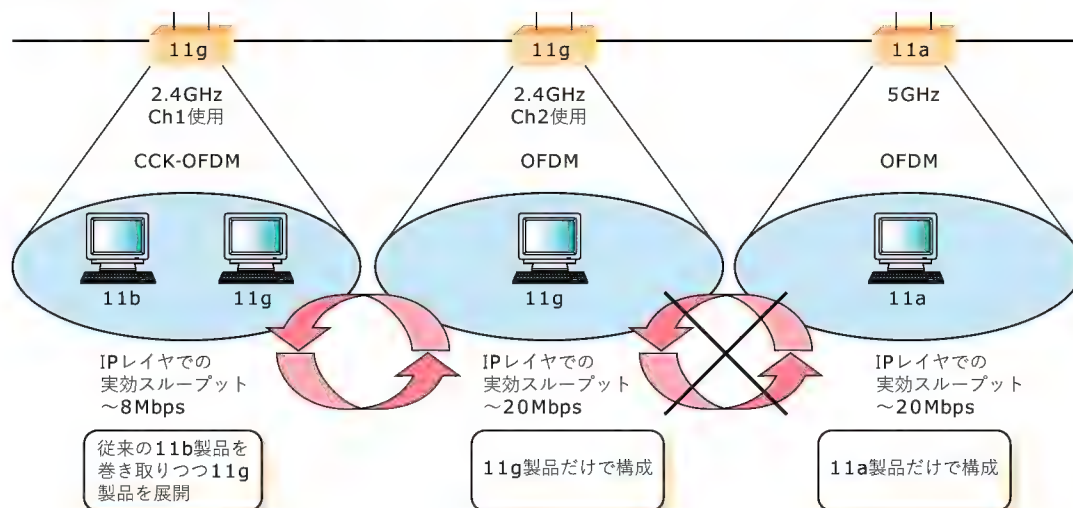
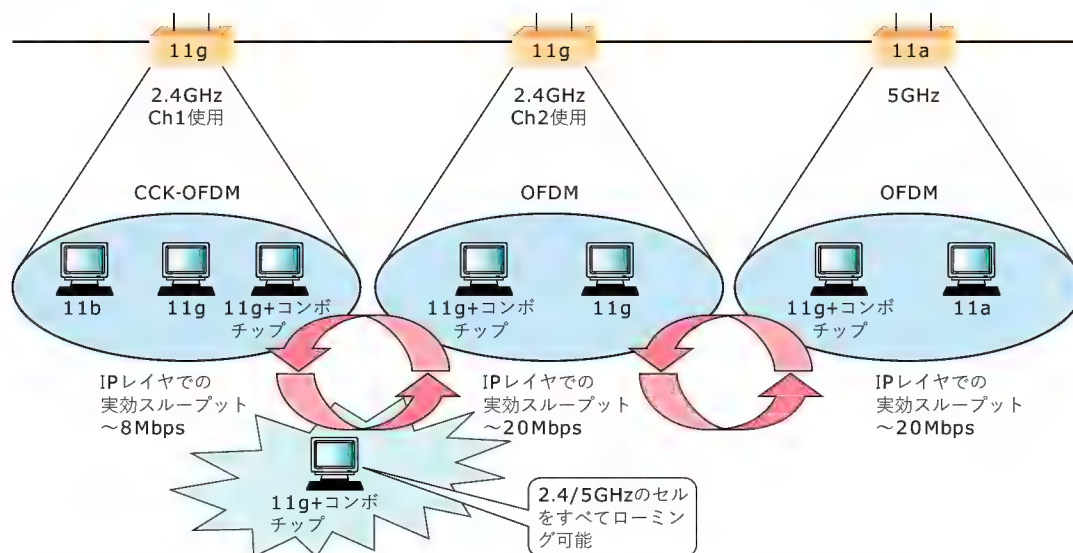
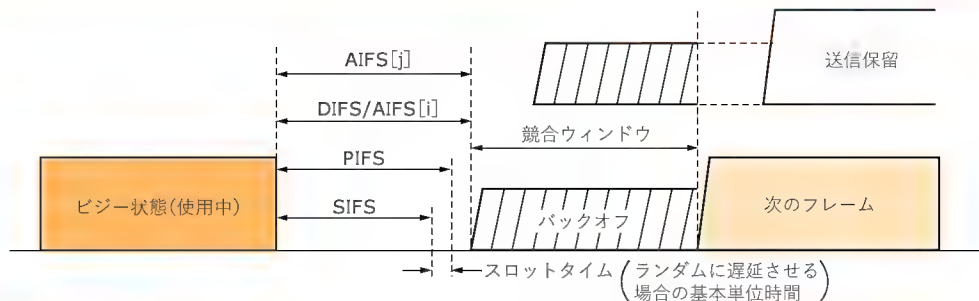
従来製品との互換性という観点からは，**図29**に示すようにAPを置き換えることで，従来のIEEE802.11，IEEE802.11bの両規格の製品を収容することができるようになっている．これは，**図30**に示すようにプリアンブル部およびヘッダ部は共通のDS-SS方式を用い，データ部分に高能率変調方式を適用することで実現する．しかし，従来方式との互換性を重視するあまり，上位レイヤ(IPレイヤ)におけるスループットは物理レイヤの高速化ほど伸びない．これはプリアンブル部およびヘッダ部の占める時間が相対的に長くなるためである(**図31**)．IEEE802.11gについて，3.5節と同様にIPレイヤでのスループットを求めたものを**図32**に示す．このようにCCK-OFDM方式では，物理レイヤの高速化の効果が十分に得られていない．

そこで，従来方式との後方互換性と決別して，将来システムとして有望と考えられる5GHz帯IEEE802.11a規格との互換性を考慮して規定されたものが，IEEE802.11gにおけるOFDM方式である(**図33**)．これにより，IPレイヤでの高スループットが期待できる．

このようなIEEE802.11gに準拠した製品が登場するとワイヤレスネットワーク環境はどのように変化していくかを**図34**，**図35**に示す．2.4GHz帯では，IEEE802.11gにより後方互換性を維持したネットワークと高スループットを追及したネットワークが構成可能となるが，2.4GHz帯と5GHz帯の間には，同じOFDM方式を使っていても物理的に周波数帯が異なるため相互に通信はできない状態が起こる．しかし，現在さかんに開発されている2.4GHz/5GHz帯のデュアルバンド対応のデバイス(コンボチップ)が普及すれば，**図36**のように2.4GHz帯IEEE802.11b/gと5GHz帯IEEE802.11aのシステム間をシームレスに接続可能

〔図32〕IEEE802.11gのスループット(理論限界値)

〔図33〕IEEE802.11gの前方互換性

になる．

このように，IEEE802.11gの登場により，2.4GHz帯無線LANと5GHz帯無線LANはどちらが良いのかという「競合状態」から，物理的に周波数帯は離れているものの「統合状態」へと移り，

〔図34〕2.4GHz帯IEEE802.11b/gのネットワーク進化

〔図35〕2.4/5GHz帯IEEE802.11a/b/g混在ネットワーク

〔図36〕2.4/5GHz帯コンボチップによるローミング

〔図37〕AIFSを用いた優先制御

AIFS	Arbitration Inter Frame Space, フレーム送信間隔
DIFS	Distributed Inter Frame Space, 分散制御用フレーム間隔
PIFS	Point Inter Frame Space, ポーリング用フレーム間隔
SIFS	Short Inter Frame Space, 短フレーム間隔
競合ウィンドウ	各パケットの衝突を避けるために設けられたランダムな遅延時間のエリア
バックオフ	各パケットでは，ビジー状態から送信に移る際にランダムな遅延時間を生成する．この時間のこと

相互に使用できるチャネルが増加したワイヤレスネットワーク環境が実現される．

4.2 タスクグループe(TGe)：QoSサポート

TGeでは，既存のMACレイヤを拡張し，QoS(Quality of Service)をサポートするための手段として，従来のDCFとPCFの機能を統合したHCF(Hybrid Coordination Function)の機能について検討が進んでいる．HCFにはQoSサポートのため，提供するQoSの性質に応じて2種類のアクセス制御方法の規定が盛り込まれる見通しである．

1) HCF競合チャネルアクセス (HCF contention-based channel access)

CSMA/CA方式を拡張したアクセス制御方式で，データ送信時に優先制御を行うプライオリティベースのQoSである．帯域や遅延時間などの具体的な品質を保証するものではない．サポートするAPおよびSTAは優先度に対応するキューをもち，アクセス制御時のパラメータであるキャリアセンス時間の長さとバックオフアルゴリズムの乱数発生範囲を優先度に応じて変化させる(**図37**)．これによって，優先度の高いデータに対してより多くの送信機会を与えることで，統計的な優先制御を実現する．

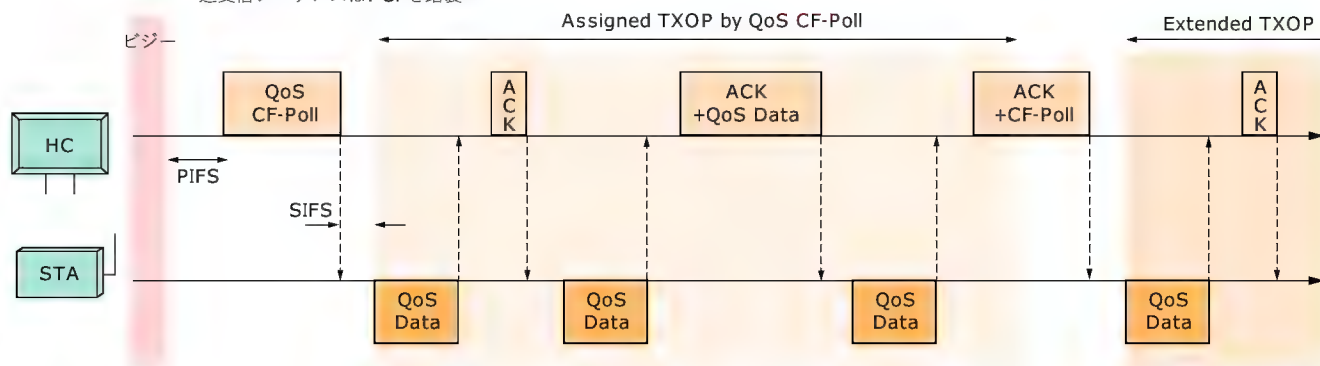
2) HCFポールド・アクセス(HCF polled channel access)

従来のポーリング手順を拡張し，指定された帯域幅や遅延時間などのパラメータを保証するためのアクセス制御方法である(**図38**)．この方法を用いるために，サポートするSTAが通信品質を要求する手順も別途規定している．

4.3 タスクグループi(TGi)：セキュリティ機能拡張

TGiでは，セキュリティ機能の拡張について，上位レイヤ認証機能と暗号化アルゴリズムの強化が検討されている．TGiは，従来のIEEE802.11になかったセキュリティ機能を提供するネットワークとしてRSN(Robust Security Network)を規定しており，その定義は「認証と鍵配送にIEEE802.1x規格を使用したネットワーク」とされている．

RSN内には認証サーバが存在し，TGi規格に準拠したAPお

〔図38〕HCFにおけるポールド・アクセス制御

よびSTAはIEEE802.11プロトコル上にIEEE802.1xを実装する．これにより，IEEE802.1xを使った認証やAP-STA対ごとに固有な暗号化鍵の使用や動的な更新などの機能が利用できる．

1）上位レイヤ認証機能

TGiで検討中の規格では，従来のIEEE802.11におけるMACレイヤ認証処理は行わず，代わりに上位レイヤ認証を採用している．認証方式に特定な方式を指定してはいないが，認証に使用するしくみとして，IEEE802.1xに規定のEAPOL（Extensible Authentication Protocol over LAN）を想定している．認証方式としては，Windows XP標準サポートのIEEE802.1xを利用したEAP-TLS（Extensible Authentication Protocol-Transport Layer Security）が主流になると見られる（**図39**）．

2）暗号化機能

WEPに代わる暗号化方式として，2種類の方式を規定しようとしている．一つは，TKIP（Temporal Key Integrity Protocol）と呼ばれ，既存のハードウェア（RC4暗号）を使用しつつ安全性を高めることを目的としたものである．TKIPでは，パケットごとの暗号鍵の導出や改ざん防止機能の強化などが考慮されている．TKIPは，すでに出荷済みの製品に対する暗号化の強化策という位置付けであり，将来的には次に紹介するAES（Advanced Encryption Standard）に移行するものと考えられている．

AESは，現時点で解読されていない非常に強力な暗号化方式であり，AESを使用した方法は，RSNにおけるデフォルトの暗号化方式となっている．AESの動作モードとしては，CCM（Counter mode with CBC MAC）方式が提案されており，必須方式になる予定である．

4.3 その他の動き

IEEE802.11ワーキンググループでは，その他にも，タスクグループf（TGf），タスクグループh（TGh），HT SG（High Throughput Study Group），WNG SC（Wireless Next Generation Standing Committee）などが活動している．

TGfは，AP間通信プロトコルIAPP（Inter-Access Point Protocol）に関する標準化を行った．IAPPは，同一ネットワークに存在するAP間でSTAのモビリティをサポートするプロトコルである．TGfは2002年5月にドラフトを完成し，ワーキンググループの承認を得ている．

TGhは，5GHz帯無線LANに対する欧州の周波数規則への対応を目的としている．具体的には，IEEE802.11a規格に送信電力制御機能（TPC：Transmission Power Control）と，動的周波数選択機能（DFS：Dynamic Frequency Selection）を追加することが検討され，2002年10月ドラフトがワーキンググループに承認されている．

HT SGは，WNG SCから分離独立したスタディグループで，現在，次期高速無線LANの規格検討に向けて準備中である．基本的に既存のIEEE802.11系の拡張による高スループット化を目的としている．

WNG SCでは，将来の無線LANシステムについて，世界統一規格の検討を目的の一つとしており，その要求条件の抽出やシステム・コンセプトなどの基礎的な議論が行われている．

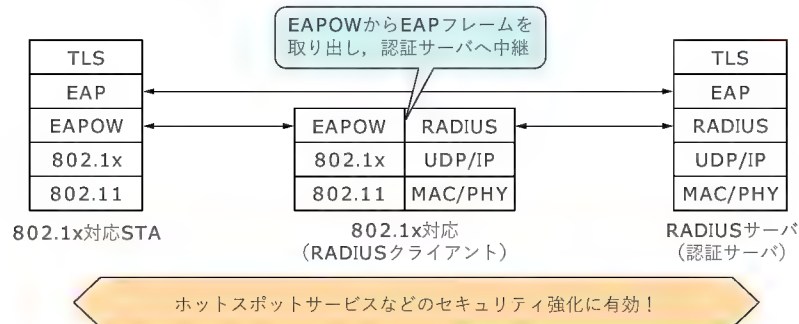
おわりに

ワイヤレスネットワーク技術の現況と題して，現在もっとも普及しているIEEE802.11系無線LANシステムの技術，標準化の動きを中心に解説した．

ワイヤレスネットワークに望まれる将来的な方向としては，以下の2点がとくに重要であると考える．

1）さらなるブロードバンド化：単なる「瞬間最大風速」としてではなく，実用的なエリアでの高速化が重要となる．無線の周波数といった限られた資源を有効に使っていく技術が必須である

2）セキュリティの高機能化：無線である性格上，セキュリティ面での課題を常に有しているといえる．ビジネスシーンやイ

〔図39〕上位レイヤ認証機能

ンフラ系のサービスにはもちろん，家庭においても発熱機器や防犯システムなどへの適用を考慮すると，セキュリティ確保は重要な課題である

＊　　＊

無線LANは，PCの普及，機器の低価格化，ブロードバンド化の波とともにビジネスシーンから家庭の中へと一気に普及してきた．今後，家電製品やAV機器を含めた家庭内ネットワークの媒体としておおいに期待されるところである．

また一方では，公衆無線LANサービスなどのネットワークサービスへの適用が進んできている．LANの技術が，ネットワークインフラへ展開されてきたのは，有線系ネットワークと同様の流れであるとともに，無線の有する「移動性」を活かした「ユビキタスネットワーク」の媒体として期待されている．

参考文献

1) 服部 武，岡 雅宣編著，『ワイヤレス・ブロードバンド教科書』，IDGジャパン
2) 日経コミュニケーション・日経ニューメディア共編，『最新モバイル・インターネット徹底解剖』，日経BP

さかた・てつ　NTTアクセスサービスシステム研究所

Chapter 2

Linux 上で動作する無線通信システムを構築する

Bluetoothプロトコルスタックの開発と検証

● 中野敬仁

近距離無線通信規格 Bluetooth が話題になっている．本章では，Linux 上で動作する Bluetooth プロトコルスタック「At-BT」および，ARM7 CPU ボード「Armadillo」上で「At-BT」を評価するキット「At-BT-EVA」について，その詳細と使用例を解説する〔いずれも開発は（株）アットマークテクノ〕．また，実際に Bluetooth 対応製品を開発したうえでの問題点も考察する．

（編集部）

はじめに

本章では，Linux 対応 Bluetooth プロトコルスタック「At-BT」，そしてその評価キットである「At-BT-EVA」についての解説と，実際にサンプルアプリケーションを作成した例を示しながら解説します．

● Bluetooth とは

Bluetooth は，エリクソン，IBM，インテル，ノキア，東芝の 5 社が中心となって策定した近距離無線の通信規格です．特徴として，次のような点があげられます．

- データ転送レートは 1Mbps
- 接続距離は 10m ～ 100m 程度
- データ（非同期データ），および音声（同期データ）が標準で利用可能
- 低消費電力
- 1 台～ 7 台での接続が可能
- 仕様が公開されている
- ISM 帯（2.402GHz ～ 2.480GHz）を利用しており，免許が不要
- 電波を使用するため無指向
- セキュリティに関する機能が標準で利用可能

「あらゆるモバイル端末同士をワイヤレスで接続する」という，Bluetooth の位置付けを元に，上記のような各パラメータが最適化されています．

とくにモバイル端末を利用する上では重要な「消費電力」について，仕様が明言されている点は特徴的です．

また，接続の形態として単に 1 対 1 の機器同士だけでなく，1 台の機器（マスタ： master）に対して複数台数の機器（スレーブ： slave）が接続可能である点は，Bluetooth が柔軟なネットワーク形成のためのインターフェースとして利用できることを示しています．

● プロトコルスタック概要

Bluetooth は，「Core」と「Profile」という 2 種類の技術仕様書によって規格が定義されています．

「Core」では，ベースバンドレイヤにおける通信方式や，送受信される際のデータフレームの定義など，各プロトコルレイヤの機能動作についての規定が記述されています．「Profile」では必要な機能動作がアプリケーション（FAX，LAN など）ごとに定義されています．

Core および Profile の詳細については，ここでは触れません．詳細については，Bluetooth SIG の公式サイトで公開されている仕様書をご覧ください．

Linux に対応した Bluetooth プロトコルスタックの評価キット「At-BT-EVA」とは

At-BT-EVA の概要図を**図 1** に示します．At-BT-EVA は，次のような構成になっています．

● Armadillo

ARM7 系の CPU を搭載，Linux を標準 OS とする組み込み向け小型 CPU ボードです．

● BT 基板

Bluetooth ベースバンドモジュール，および Armadillo と接続するためのインターフェースを含む接続基板です．

モジュールには NBTC-1 シリーズ〔NBTC-119A ：長野日本無線（株）〕を使用しています．BT 基板の接続には，Armadillo の COM2［ttyAM1］を使用します（**写真 1**）．

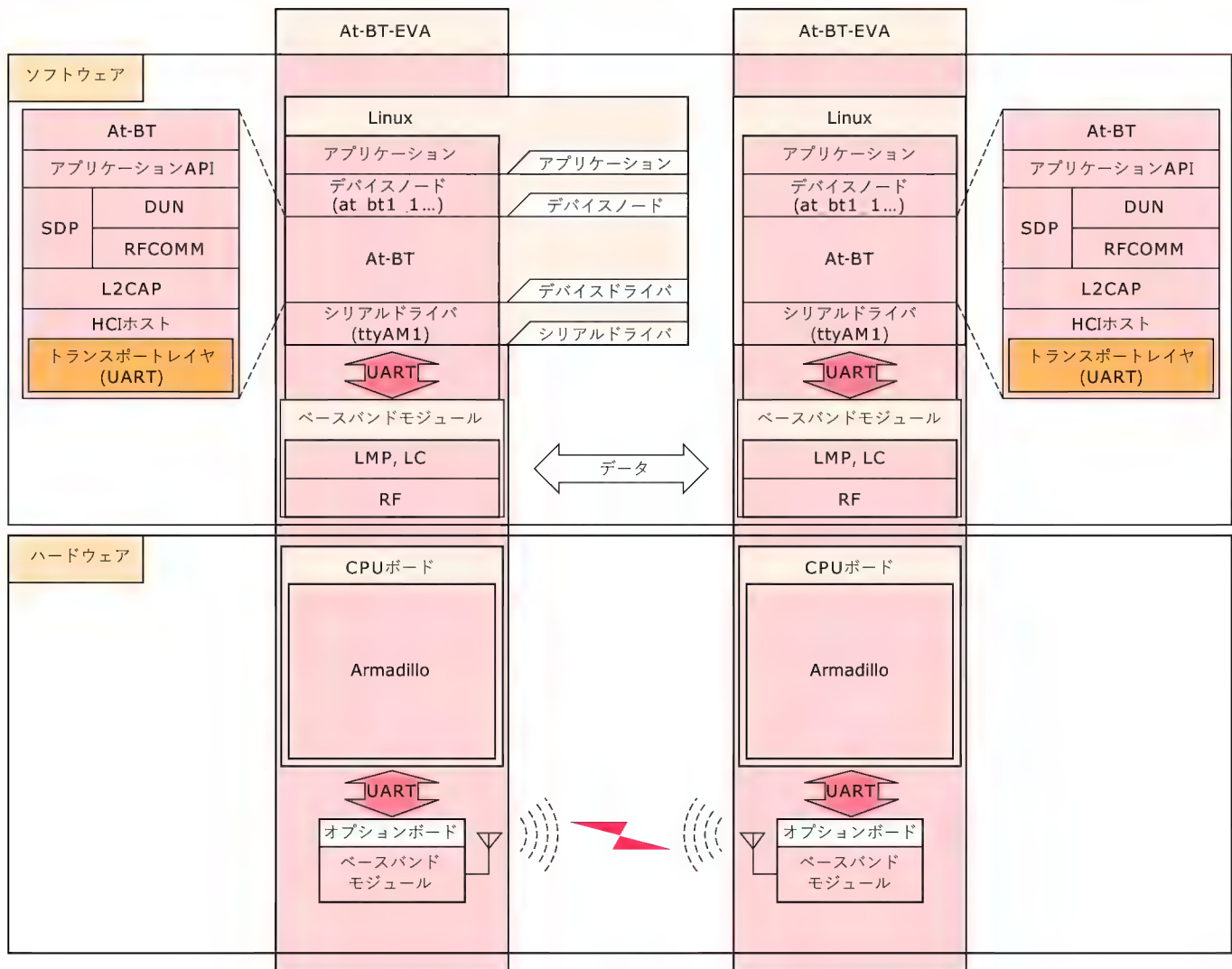
● At-BT

Linux 対応の Bluetooth プロトコルスタックです．Armadillo 専用のオブジェクトファイルとして提供されます．

「At-BT-EVA」を使用することで，Bluetooth を使った無線通信が手軽に行えます．また，API を使用した At-BT を制御するプログラムを作成することで，Bluetooth に対応したアプリケーションの開発が可能です．

At-BT-EVA の特徴を次に示します．

〔図1〕At-BT-EVA概要図

At-BT-EVA
At-BT-EVA
ソフトウェア
At-BT
アプリケーションAPI
SDP
DUN
RFCOMM
L2CAP
HCIホスト
トランスポートレイヤ (UART)
Linux
アプリケーション
デバイスノード (at_bt1_1…)
At-BT
シリアルドライバ (ttyAM1)
UART
ベースバンドモジュール
LMP, LC
RF
アプリケーション
デバイスノード
デバイスドライバ
シリアルドライバ
データ
ハードウェア
CPUボード
Armadillo
UART
オプションボード
ベースバンドモジュール

●Linux対応Bluetoothプロトコルスタック「At-BT」が動作
●Version1.1のロゴ認証取得済み(At-BT, At-BT-EVA)
●HCIトランスポートレイヤにUARTを使用
●既存のLinuxシリアルアプリケーションが使用可能
●ピコネット対応
●プロトコルスタック自体をttyデバイスドライバとして実装

1.1 プロトコルスタック「At-BT」とは？

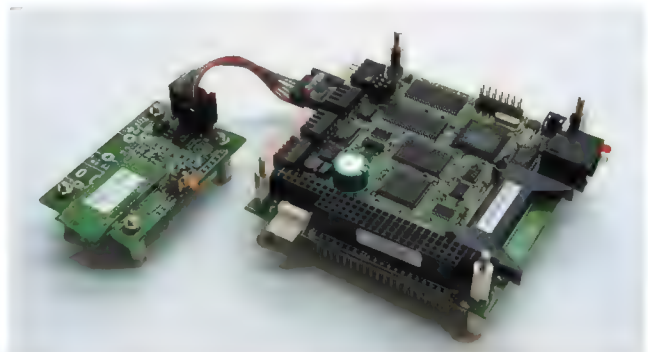
プロトコルスタック「At-BT」について説明します.

● 対応プロトコルとプロファイル

現在対応しているプロトコルとプロファイルは，**表1**，**表2**のようになっています．**表2**のプロファイルのうち(＊)マークに関しては，2001年12月にBluetooth SIGよりロゴ認証を取得しました．

● スタックレイヤ構成

At-BTは，次のようなレイヤ構成になっています(**図2**)．At-BTはデバイスドライバとして実装されています．

〔写真1〕Armadilloと BT基板を接続したようす

▶ Bluetoothプロトコルスタックレイヤ

Bluetooth Specification Version 1.1 Core，およびProfileの仕様にしたがい，次のレイヤが実装されています．

●HCI-UART(トランスポートレイヤ)

ホスト(プロトコルスタック)とホストコントローラ(ベースバ

コラム BNEP

Bluetooth標準のTCP/IP機能を実現するプロトコルとして，BNEP（Bluetooth Network Encapsulation Protocol）があります．BNEPを使用することで，RFCOMMとDUNP（Dial-up Networking Profile）を使用した場合に比べ，通信の際のオーバヘッドが軽減されるという利点もあります．

BNEP自体のVersionが0.9xxということでまだまだ策定中の段階なのですが，近年，PAN（Personal Area Network）に対する要求は高まってきているため，現在のRFCOMM＋DUNPの構成は，PANのプロファイルにて必須とされるBNEPにいずれ置き換わることでしょう．

ンドモジュール）間において，データの送受信をどのように行うかについて取り決めを行っているプロトコルです．

「UART・PCMCIA・USB」のインターフェースが定義されています．At-BT-EVAでは，UARTを使用しています．転送レートは115.2kbpsで，RTS/CTSによるフロー制御は行っていません．

● HCI

ホスト-ホストコントローラ間での制御方法が定義されているプロトコルです．ホストが送信する「コマンド」と，要求に対するホストコントローラからの「イベント」が規定されています．

At-BTでは，複数コネクションによるデータ送受信に対応しています．また，送信コマンドと受信イベントの関連付け用管理テーブルをスタック内部に設けて同期処理を行っています（**図3**）．これにより，一つの機器に複数のモジュールをつないで利用される状況において，送信コマンドと受信イベントの同期（関連付け）処理が可能になります．

● L2CAP

「論理リンクチャネルの制御」，「データの分割・再構成」などの処理を行うためのプロトコルです．複数の論理リンクチャネルでの動作に対応しています．L2CAPの上位のスタックはSDPやRFCOMM，BNEP，AVCTP（Audio Video Control Transport Protocol）などというように，各種の適合プロトコルが当初から規定されています．

At-BTでは，L2CAPとその上位のプロトコル間のインターフェースについて，上位に位置する各種のプロトコルの変更（取りかえられて使用されること）を前提とした構造としました．これは，**図4**に示すように「上位のプロトコルの情報をL2CAPレイヤに伝える（レジストする）」という処理を行うことで実現しています．

● SDP

Bluetooth機器として利用可能なサービスの検索・応答機能を実現するためのプロトコルです．対応プロファイルとして登録しているサービスは，次のようになっています．

- Serial Port Profile
- Service Discovery Application Profile

At-BTではクライアント（サービス問い合わせを行う）側，およびサーバ（サービス問い合わせに対しての応答）側の両機能に対応しています．

今回，At-BTで対応しているプロトコルのうちでも，SDPは次のような点において特殊であるといえます．

- ビッグエンディアン形式でデータを扱う（他のレイヤはリトルエンディアン）
- データベース（検索・応答）処理

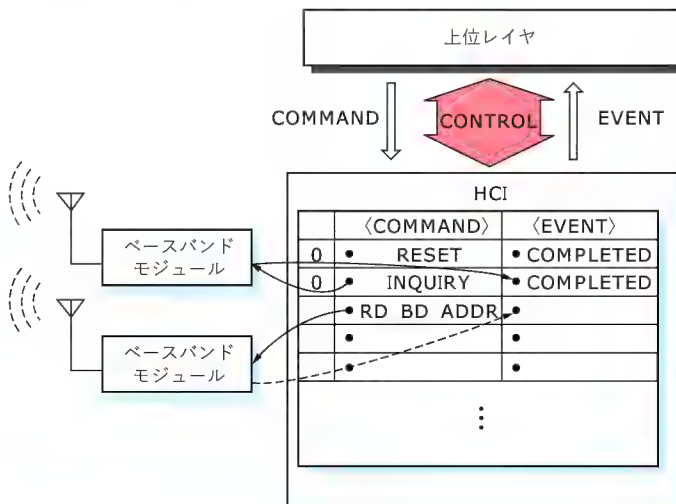
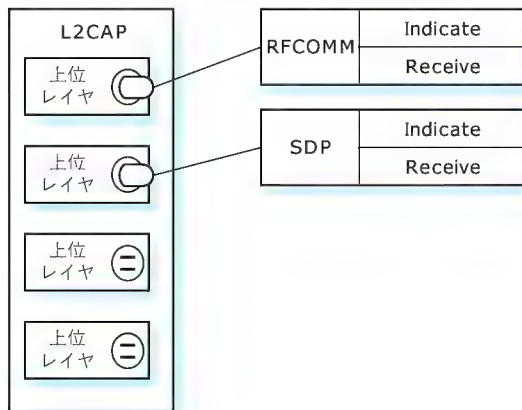
〔表1〕対応プロトコル一覧

L2CAP（Logic Link Control and Adaptation Protocol）
RFCOMM
SDP（Service Discovery Protocol）
HCI-UART

〔表2〕対応プロファイル

GAP（Generic Access Profile）[＊]
SDAP（Service Discovery Application Profile）[＊]
SPP（Serial Port Profile）[＊]
DUNP（Dial-Up Networking Profile）

〔図2〕At-BTスタックレイヤ構成図

〔図3〕HCI送信コマンド，受信イベントの同期処理

〔図4〕L2CAP レイヤへ上位レイヤ関数を登録

一般的な通信プロトコルの基本動作である「上位レイヤからのデータにヘッダ情報を付加して下位レイヤに渡す」という処理に加えて，**図5** に示すような「受信データに対する検索・対応データの送信」という処理が加わります．

このような理由により，SDP レイヤはほかのレイヤより構造が複雑です．

● RFCOMM

シリアルポート（RS-232-C 準拠）の機能をソフトウェアエミュレートするための各種設定が定義されているプロトコルです．Core Version 1.0b，および 1.1 に対応しています．

● DUNP

At-BT では各種制御が行えるように，次の機能を実現する独自の AT コマンドを実装しています．

● 接続制御用
● HCI コマンド発行用

minicom のような AT コマンドの発行が可能な，一般的なシリアルアプリケーションを使用して At-BT を制御することで，Bluetooth による通信が手軽に実現できます（**図6**）．

▶ OS アブストラクションレイヤ

At-BT は Linux を標準の対応 OS としていますが，Linux 以外の他の OS への移植が容易に行えるように，使用している OS のシステムコールをカプセリングする構造をとりました．また，プロトコルスタックの各レイヤがよく利用する関数（文字列の比較など）をこのレイヤに独自に登録しておくことで，OS への依存度を低減させています．

とくに近年は Linux の開発進度が速く，カーネルのバージョンによってシステムコールのインターフェースが異なることもたびたびあります．このレイヤの機能を利用することで，At-BT が複数バージョンの Linux カーネルで動作することを可能にしています．

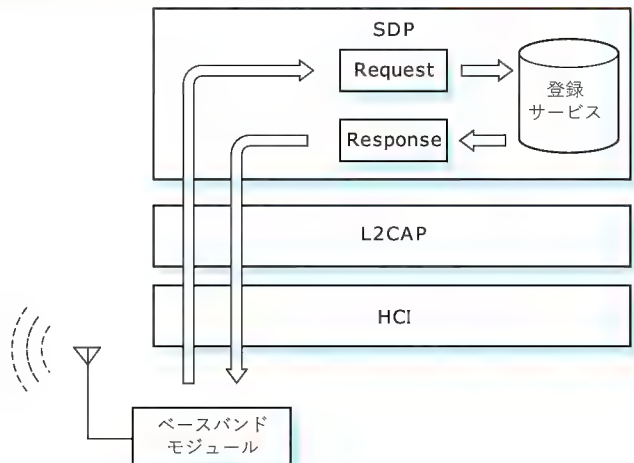
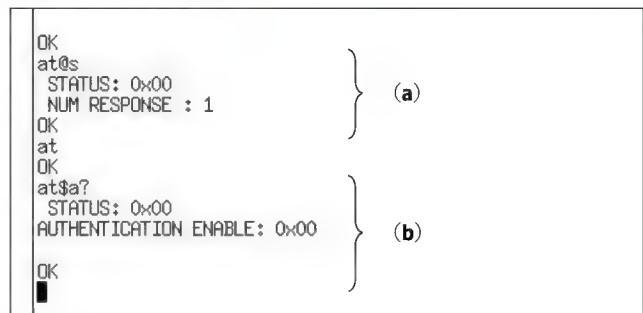
▶ Linux ドライバレイヤ

At-BT を制御・操作するための Linux キャラクタ型デバイスドライバ用のシステムコールが，このレイヤに含まれています．

〔図5〕SDP データベース（検索・応答）処理

〔図6〕AT コマンド発行例

```
OK
at@s
 STATUS: 0x00
 NUM RESPONSE : 1          (a)
OK
at
OK
at$a?
 STATUS: 0x00
AUTHENTICATION ENABLE: 0x00   (b)

OK
```

At-BT とデバイスドライバ構造については次節で説明します．

● 実装上のポイント

▶ At-BT の全レイヤをデバイスドライバとして実装

At-BT では対応しているすべてのプロトコルレイヤをアプリケーションではなく，デバイスドライバとして実装し，カーネルモードで動作する構造になっています（**図7**）．

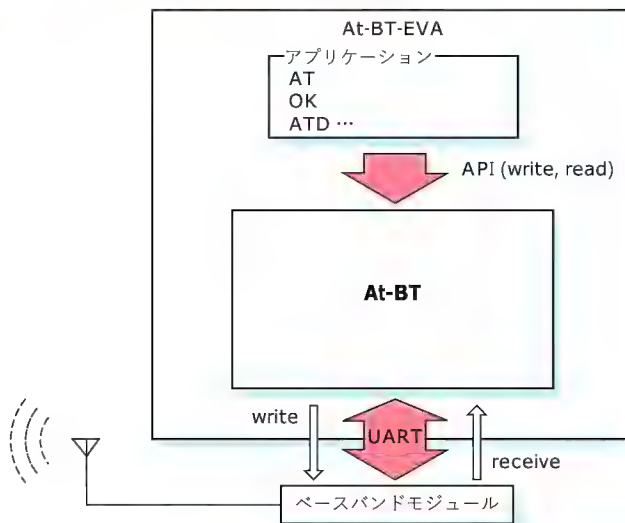
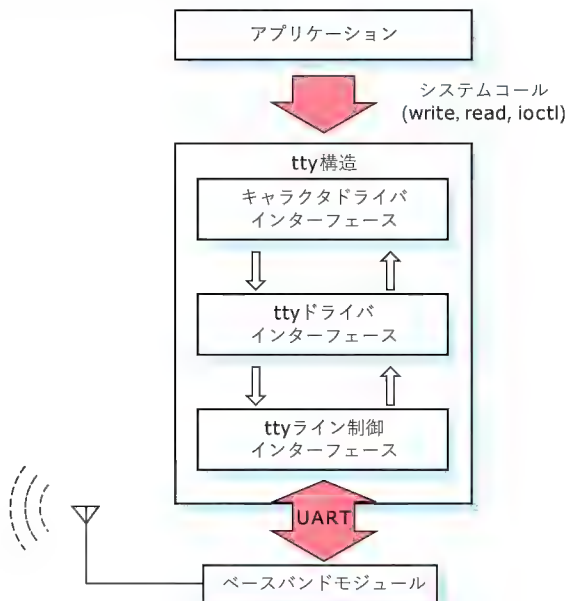
アプリケーション（ユーザーモード）としてプロトコルを動作させた場合，プロトコル処理の最中にタスクスイッチが実行されます．

その結果，プロトコル処理が中断され，パフォーマンスの低下につながります．この問題を解決するため，アプリケーションではなく，デバイスドライバとして実装するにいたりました．

▶ キャラクタデバイス（tty デバイス）として登録

At-BT は，Linux のキャラクタ型のデバイスドライバ（tty デバイス）として動作します．tty デバイスとして実装することで，次のようなメリットが得られます．

● Linux シリアルアプリケーションの資産が利用可能（後述する minicom や PPxP など）
● tty のデバイスドライバの設計思想，および資産をそのまま流用することで開発工数を軽減させる
● ファイル操作用のシステムコールを使ったプログラミングが

〔図7〕アプリケーション・At-BT・UART相関図

〔図8〕Linux ttyドライバ構造

〔表3〕ファイル解説

ファイル	詳　細
n_tty.c	ttyライン制御(UARTに依存する制御)に関する処理
serial_clps711x.c	At-BT-EVAで使用しているCPU(clps711x系)に搭載されているUARTに依存する処理
serial_core.c	キャラクタ型デバイス用システムコールがコールされた際に実行される処理
tty_io.c	コンソール，UARTなどのttyデバイスの内，使用するデバイスに依存しない共通処理
tty_ioctl.c	ttyドライバ，およびttyライン制御のうち「ioctl」に関する処理

簡単に行える

ここで，ttyの構造について説明します．

● tty構造

Linuxのtty型のデバイスドライバは，三つのレイヤに分かれています(**図8**，**表3**)．

At-BTは，このttyの上位2レイヤを置き換える形で実装されています．3レイヤに分けられたttyのドライバは，各レイヤの置き換えが可能であり，独自のttyドライバを作成できる思想をうたっています．しかし実際には，明確にレイヤ分けがされていないことや，レイヤを置き換えるために必要なシステムコールが整備されていないなど，まだ発展途上であるという感はいなめません．

そのためttyの構造に相当熟知していなければ，内部の構造に手を加える作業にかなりの工数が必要となります．

実装されたAt-BTは**表4**，および**リスト1**のように確認することができます．

動作させるにあたっていちばんの懸念は「既存のttyレイヤをBluetoothのレイヤに置き換えることによるオーバヘッドの増加」でした．実際に行ったテストの結果から，オーバヘッドの増加による影響はなく，ftpを使用したファイル転送を行った場合でも，UART設定の限界値に近い転送レートが確認できています(詳細は，次項で解説する)．

● ioctl関数を利用することでHCIの全コマンドが送信可能

キャラクタ型のデバイスドライバとして実装し，ファイル操作用システムコールが使えるようになることの大きなメリットとして「ioctl関数が利用可能である」点があげられます．ioctl関数はデバイスドライバからハードウェア制御を行うために，デバイス依存の制御コマンドを発行するためのシステムコールです．Linuxでは一般的なシステムコールで，多種多様なデバイスドライバで実装されています．

At-BTではこの「ioctl」を使用することで，すべてのHCIコマンドが送信可能です．これによりアプリケーションからベースバンドモジュールの制御が行えます．さらに，ioctl関数はデバイスドライバの開発者によって自由に拡張してよいため，At-BTに対して独自のプロトコルを実装・拡張し，ioctlを使用して動作させることも可能です．

〔表4〕デバイスノード概要

ファイル名	at_bt1_1(～at_bt1_4)
デバイス名	at_bt
メジャー番号	241
マイナ番号	17

〔リスト1〕At-BTデバイスノード設定

```
# ls -la at_bt*
crw-rw-rw-    1 root     tty      241,  17 Jan  8 13:00 at_bt1_1
crw-rw-rw-    1 root     tty      241,  17 Jan  8 13:00 at_bt1_2
crw-rw-rw-    1 root     tty      241,  17 Jan  8 13:00 at_bt1_3
crw-rw-rw-    1 root     tty      241,  17 Jan  8 13:00 at_bt1_4
```

〔リスト 2〕**minicom 設定ファイル**(minirc.dfl)**例**

```
pr port          /dev/at bt1 1
pu minit         ~^M~ATZ^M~
pu mreset        ~^M~ATZ^M~
```

2 At-BT-EVA の使用例

本項では次のような例を示して，At-BT-EVA についての説明を行います．

- シリアルアプリケーション(minicom)での使用例
- シリアルデータ送受信プログラムの作成
- チャットアプリケーションの作成
- PPP(TCP/IP)機能を利用

At-BT-EVA 上で動作するプログラムの作成には，ARM オブジェクトを生成するためのクロスコンパイル環境が必要になります．環境の構築については Armadillo 公式サイト(http://armadillo.atmark-techno.com/)をご覧ください．

2.1 シリアルアプリケーションでの使用例

At-BT-EVA の特徴である「AT コマンドを使用して簡単に接続が可能である」点について，シリアルアプリケーションである minicom を使って接続する例を取り上げます．minicom は，多くの Linux ディストリビューションに同梱されているシリアルデバイスを制御するための通信プログラムです．

はじめに，minicom で At-BT を操作するために設定ファイルを調整します．minicom で使用する際の設定は非常に簡単で，minicom の設定ファイル内に使用するデバイスノードを指定するだけです(**リスト 2**)．

```
pr port          /dev/at_bt1_1
```

設定完了後，次の手順で操作を行います．

a) minicom 起動

まず，minicom を接続要求側(以降，リクエスト側)，被接続側(以降，レスポンス側)の 2 台，それぞれ起動させます(**図 9**)．

(シェルコマンド)

```
$ minicom
```

b) 周囲の Bluetooth 対応機器の検出〔**図 10**(**a**)〕

リクエスト側にて検出動作を行うコマンド(at@s)を入力し，周囲に存在する Bluetooth 対応機器の検出を開始します．

(AT コマンド)

```
at@s
```

Bluetooth 機器の検出が失敗した場合は，検出件数(NUM RESPONSE)が「0」で表示されます．その際は再度検出動作を行ってください．

c) 検出結果の表示〔**図 10**(**b**)〕

検出動作後，検出が完了した機器のアドレス(「Bluetooth Device Address」，以降「BD_ADDR」)を表示させせます．

(AT コマンド)

```
at@l
```

〔図 9〕**minicom の起動**

```
Welcome to minicom 1.83.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Jan 22 2002, 14:46:14.

Press CTRL-A Z for help on special keys


OK
ATZ
OK
```

〔図 10〕**minicom**(リクエスト側)

d) 接続開始・接続完了

c)の結果より接続先を指定し，接続動作を開始します．

(AT コマンド)

```
atds=0
```

接続が完了した場合「CONNECT」と表示されます〔**図 10**(**c**)〕．

また，レスポンス側にも接続が完了したことを通知する文字「CONNECTED」が表示されます〔**図 11**(**a**)〕．レスポンス側は，デフォルトで「対向からの接続をすべて受け付ける設定」になっています(この設定については自由に変更が可能)．

e) データの送受信が行える

キーボードから入力を行います．入力はすべて相手側の画面に表示されます〔**図 10**(**d**)〕．

f) 切断処理

切断処理は，エスケープシーケンス(+++)を入力し，データモードからコマンドモードに移行します(接続完了後は自動的にデータモードになっており，コマンドの入力は行えない状態になっている)．その後，回線切断の AT コマンド(ath)を入力します．

(AT コマンド)

```
ath
```

回線切断処理が完了した場合「OK」が表示されます〔**図 10**(**e**)〕．

対向側により切断処理が行われ，その処理が完了した場合

〔図11〕minicom(レスポンス側)

「DISCONNECTED」と表示されて切断処理が完了します〔**図11**(**b**)〕.

2.2 シリアルデータ送受信プログラムの作成

At-BTのAPIの利用して,実際にサンプルアプリケーションを作成します.2台のAt-BT-EVAをBluetoothで接続し,双方の画面にて入力されたデータが対向の画面に出力されるサンプルアプリケーションを作成します.アプリケーションの作成にあたっては,次のようなポイントがあげられます.

a) 接続先のBD_ADDRの指定

〔リスト3〕sample.c(抜粋)

```
  /* ## リクエスト側処理 ## */
static int RequestProc( unsigned char *target )
{

    const unsigned char AT_CONNECT_A[] = "atd00:22:44:66:88:AA\r" ;
    const unsigned char AT_CONNECT_B[] = "atd11:33:55:77:99:BB\r" ;

      /* チャネル確立動作は n回までリトライする */
    unsigned char connect_challange_count = 1 ;
    unsigned char count = 0 ;

    unsigned char *dst = NULL ;

    if( strcmp( target, "ADDR_A" ) == 0 ) {
           /* ADDR_Aへ接続 */
         dst = (unsigned char *)AT_CONNECT_A ;

    } else if( strcmp( target, "ADDR_B" ) == 0 ) {
           /* ADDR_Bへ接続 */
         dst = (unsigned char *)AT_CONNECT_B ;

    } else {
           /* 引数省略時には,ADDR_Aへ接続 */
         dst = (unsigned char *)AT_CONNECT_A ;

    }

      /* 対向機とのコネクション確立開始(atコマンド送信) */
    for( count = 0 ; count < connect_challange_count
                         && !connect_flag ; count++ ) {
         write( fds, dst, strlen( dst ) ) ;
    }

      /* リトライ回数分送信したが,チャネル確立ならず */
    if( !connect_flag ) {
         return FAILED ;
    }

    return COMPLETED ;

} /* RequestProc */
```

b) 接続・切断処理にATコマンドを使用

c) Linuxのキャラクタ型デバイスドライバ用のシステムコールを使用

事前に接続対象のBD_ADDRを設定しておきます.

```
const unsigned char AT_CONNECT_A[] =
                     "atd00:22:44:66:88:AA\r" ;
```

サンプルプログラム中では,接続するBD_ADDRをあらかじめ指定して接続を行っていますが,初めに機器の検出(inquiryを発行するAPIを利用)をしておき,検出された機器に対して接続を開始することも可能です.

また,接続・切断などの制御もATコマンドで行うので,プログラム内で設定しておきます.

「open, select, write」といった,キャラクタ型デバイスドライバのシステムコールを使用してデバイスの制御を行います.

```
  /* ADDR_Aへ接続 */
dst = (unsigned char *)AT_CONNECT_A ;
            …
            略
            …
write( fds, dst, strlen( dst ) ) ;
```

実際に作成したシリアルデータ送受信プログラムのソースコード(sample.c)の一部を**リスト3**に示します.

2.3 チャットアプリケーションの作成

前節のシリアルデータ送受信プログラムをベースに,1対1で行うチャットアプリケーションを作成します.At-BT-EVAを使う場合,シリアルコンソール(もしくはEthernet経由)からログインし,シェルよりコマンドを打ち込んで行う「キャラクタベースでの操作」になります.そのため,本節で作成するチャットアプリケーションもキャラクタベースになります.今回のアプリケーションを作る上で「curses」という画面制御用のライブラリを使用します.

ここで,cursesについて簡単に説明します.cursesはキャラクタベースの対話型アプリケーションを作成するためのライブラリで,LinuxをはじめとするUNIXベースシステムでは標準的なライブラリです.Linuxを標準OSとするArmadilloにおいても,問題なく利用することができます.今回作成したチャットの画面イメージを**図12**に示します.

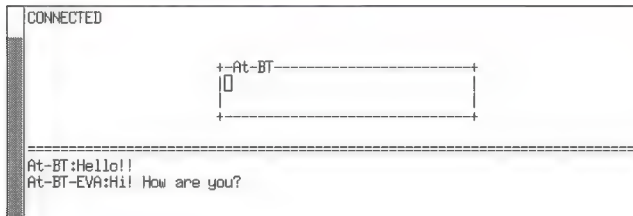
チャット中は発言がどちら(誰)からのものであるか判別するために,事前にお互いの名前を入力しておく必要があります.チャットの開始時に自分の名前を伝えて(もしくは相手の名前を確認して)おきます.

この処理を行うために,Bluetooth(HCI)の機能であるChange_Local_Name /Remote_Name_Requestコマンドを使用します.Change_Local_Nameコマンドは,Bluetoothベースバンドモジュールに対して248バイトのデータ(Name)を登録しておくことができる機能です.Remote_Name_Requestコマンドは,コネクション確立後に相手のNameを取得できるという機

〔図12〕チャット画面動作イメージ

```
CONNECTED

              +-At-BT--------------------------+
              |□                               |
              |                                |
              +--------------------------------+

==========================================================
At-BT:Hello!!
At-BT-EVA:Hi! How are you?
```

〔表5〕アプリケーション作成に必要なファイル

ファイル名	chat_sample.c
	my_chat.c
	sample.h
	my_chat.h

能です(チャットにはうってつけの機能だといえる).

アプリケーションよりHCIのRemote_Name_Requestコマンドを発行するためは,**リスト4**のようにします.

今回作成するチャットアプリケーションは,次のファイルから構成されます(**表5**).以上のファイルを元にコンパイルを行い,アプリケーションを作成します.

(シェルコマンド)

```
$ arm-linux-gcc -Wall -o smpl_chat -I./
    include -I/usr/arm-linux/include -I/usr/
    arm-linux/include/ncurse/  chat_sample.
    c my_chat.c -lncurses -mstructure-size
                                   -boundary=8
```

上記のコンパイルオプションについて,ポイントが2点あります.

- 構造体のサイズ調整方式の指定

```
-mstructure-size-boundary=8
```

構造体,および共用体のサイズ調整(丸め)が発生する際,その動作方法を指定します.At-BTを利用する際には必ず「8」を指定してください.

- cursesライブラリ利用の指定

```
-lncurses
-I/usr/arm-linux/include/ncurse/
```

cursesライブラリを使用する際のオプション指定です.`-I`オプションに関しては,個々の環境にあわせて指定を行ってください(筆者の環境では「`/usr/arm-linux/include/ncurse/`」にcursesライブラリ用のヘッダファイルが存在する).

上記のコマンドにて作成されたファイル「`smpl_chat`」をAt-BT-EVA上で実行します.

a) レスポンス側を起動

はじめにレスポンス側のAt-BT-EVAにて作成したプログラムを起動させます.引き数を指定する必要はありません.

(シェルコマンド)

```
$ smpl_chat
```

〔リスト4〕chat_sample.c(抜粋)

```
  /* ## 対向側のName取得 ## */
static int GetAndSetOppoNm( struct cmn_bd_addr *bd_addr )
{

    int result = 0 ;
    struct remote_nm_req_prm prms_nm ;

      /* 指定のBD_ADDRの機器名取得(Remote Local Nameコマンド発行) */
    prms_nm.set_bd_addr = bd_addr ;
    prms_nm.page_scan_repetition_mode =
                                    HCI_PAGE_SCAN_REPETITION_MODE_R1 ;
    prms_nm.page_scan_mode = HCI_PAGE_SCAN_MODE_MANDATORY ;
    if( (result = ioctl( fds, BTCTL_HCI_REMOTE_NM_REQ,
                                              &prms_nm ) ) < 0 ) {
        return FAILED ;

    }

      /* ローカルに保存 */
    SetOppoNm( prms_nm.remote_nm ) ;
    return COMPLETED ;

} /* GetAndSetOppoNm */
```

b) リクエスト側を起動

続いて,リクエスト側にて起動させます.

(シェルコマンド)

```
$ smpl_chat ADDR_A
```

第1引き数は,リクエスト側機器が接続動作を行う際の,相手側のBD_ADDRを指定するための引き数です.今回の場合は,上記a)で起動させた接続先(レスポンス側)のBD_ADDRが「00:22:44:66:88:AA」であるため「ADDR_A」を指定します.

c) 名前の入力

リクエスト側・レスポンス側ともに名前を入力します(**図13**).

d) 接続動作の開始

リクエスト側が接続動作を開始します.

e) 接続完了・チャットの開始

接続が完了するまで数秒かかります.接続が完了した場合,画面左上の表示が「WAITING」から「CONNECTED」になり,チャットが開始されます(**図12**).

接続動作が失敗した場合は,プログラムが終了します.

f) チャットの終了

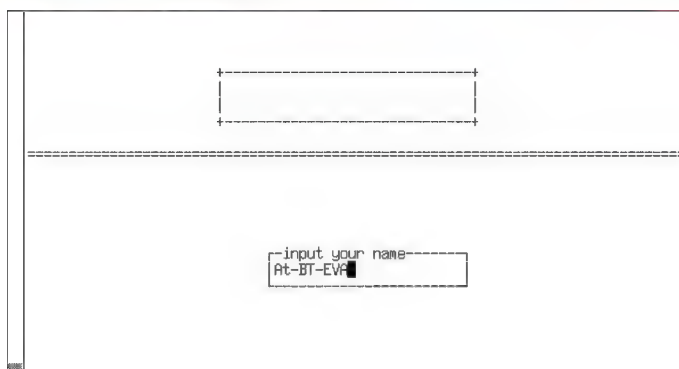
チャットを終了させる場合は,「Ctrl+Cキー」を押してアプリケーションを終了させます.

また,接続先がチャットを終了した場合は,自画面左上の表示が「CONNECTED」から「WAITING」に戻り,Bluetoothのコネクションが切断されたことがわかります.

2.4 PPP(TCP/IP)機能を利用

At-BTでは「HCI, L2CAP, SDP, RFCOMM」のレイヤがサポートされています.他の上位プロトコルに関しては,ユーザーが拡張・実装することで利用可能です.

本節で解説するTCP/IP機能を利用するためには,DUNPの仕様にしたがいPPPの機能が必要となります.幸いLinuxにはPPPフレームを生成・解釈するアプリケーションがいくつか存

〔図13〕チャット画面名前入力

在します．本節では，そのうちの一つである「PPxP」というアプリケーションを利用して実例を示します．

● PPxPについて

ここで，PPxPについて解説します．PPxPは真鍋敬士氏作のPPPドライバです．このアプリケーションを利用することにより，Bluetoothで接続されたAt-BT-EVA間でTCP/IPでの通信が可能になります．PPxPは本来，シリアルポートにモデムが接続されている機器構成が標準的なので，ttyデバイスとして利用可能なAt-BTは問題なく動作が可能です．PPxPの詳細な利用方法についてはここではふれません．PPxPの公式サイトで公開されている「PPxPガイド」をご覧ください．

〔リスト5〕設定スクリプト(myconfig)設定例(リクエスト側)

```
source qdial
set MODE active
set AUTH.PASSWD my auth pass
set LOG.FILE myconfig.log
set LINE /dev/at bt1 1
set SERIAL.MODEM /myconnect
set DIAL.LIST 0000000000
set AUTH.PROTO PAP CHAP/MD5 CHAP/MS
set IP.LOCAL 192.168.100.250/24
set IP.SLOCAL yes
set IP.VJ no
set IP.RESOLV no
```

〔リスト6〕モデムスクリプト(myconnect)例

```
include         standard

Name            "For At-BT with Hayes AT compatible "
MaxDTESpeed     115200
Initialize      "at\r\5d"
Dial            "ATD00:22:44:66:88:AA"

Ok              "OK"
Connect         "CONNECT"
Ring            "RING"
Error           "ERROR"
Busy            "BUSY"
NoCarrier       "NO CARRIER"
```

〔リスト7〕設定スクリプト(myconfig)設定例(レスポンス側)

```
source qdial
set MODE active
set AUTH.PASSWD my auth pass
set LOG.FILE myconfig.log
set LINE /dev/at bt1 1
set SERIAL.MODEM /myconnect
set DIAL.LIST 0000000000
set AUTH.PROTO PAP CHAP/MD5 CHAP/MS
set IP.LOCAL 192.168.100.200/24
set IP.SLOCAL yes
set IP.VJ no
set IP.RESOLV no
```

PPxPを使用してPPPのコネクションを確立するために，次の二つの設定が必要です．

I) PPxPが制御するデバイスにAt-BTを指定

設定スクリプト「/home/guest/.ppxp/conf/myconfig」ファイル中の使用するデバイスの項目に「At-BT」を指定します(**リスト5**)．

```
set LINE /dev/at_bt1_1
```

II) 接続先のBD_ADDRを指定

モデムスクリプト「/usr/local/etc/ppxp/modem/myconnect」ファイル中の接続(Dial)コマンドに，接続先のBD_ADDRを設定します．接続先のBD_ADDRが「00：22：44：66：88：AA」の場合は，dialのATコマンドである「ATD」に続けて，

```
Dial          "ATD00:22:44:66:88:AA"
```

と設定します(**リスト6**)．

リクエスト側のIPアドレスを次のように設定しておきます(**リスト5**)．また，レスポンス側のIPアドレスはリクエスト側のアドレスと異なる値を設定してください(**リスト7**)．

(設定例)

［リクエスト側］

```
set IP.LOCAL 192.168.100.250/24
```

［レスポンス側］

```
set IP.LOCAL 192.168.100.200/24
```

PPxPを起動させ，接続を開始します．接続が完了(ppxpコンソールの文字表示が「PPXP」に変化)するまでは，数秒かかります．接続が完了したのち，TCP/IPの接続が確立したかどうかをチェックするために，ifconfigコマンドを発行します(**図14**，**図15**)．

(シェルコマンド)

```
$ ifconfig
```

［リクエスト側］

```
tun1      Link encap:Point-Point Protocol
   inet addr:192.168.100.250
       P-t-P:192.168.100.200
       Mask:255.255.255.0
```

［レスポンス側］

```
tun1      Link encap:Point-Point Protocol
   inet addr:192.168.100.200
       P-t-P:192.168.100.250
       Mask:255.255.255.0
```

1対1の接続が完了して，リクエスト側・レスポンス側でIPアドレスが決定したことが確認できます．

さらに，リクエスト側よりpingコマンドを発行します(**図**

〔図14〕ifconfigコマンド発行結果（リクエスト側）

```
PPXP> ifconfig

tun1      Link encap:Point-Point Protocol
          inet addr:192.168.100.250  P-t-P:192.168.100.200  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

〔図15〕ifconfigコマンド発行結果（レスポンス側）

```
PPXP> ifconfig

tun1      Link encap:Point-Point Protocol
          inet addr:192.168.100.200  P-t-P:192.168.100.250  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

〔図16〕pingコマンド発行結果

```
PPXP> ping 192.168.100.250
PING 192.168.100.250 (192.168.100.250): 56 data bytes
64 bytes from 192.168.100.250: icmp_seq=0 ttl=255 time=1.6 ms
64 bytes from 192.168.100.250: icmp_seq=1 ttl=255 time=1.4 ms
64 bytes from 192.168.100.250: icmp_seq=2 ttl=255 time=1.4 ms
64 bytes from 192.168.100.250: icmp_seq=3 ttl=255 time=1.4 ms
64 bytes from 192.168.100.250: icmp_seq=4 ttl=255 time=1.4 ms
64 bytes from 192.168.100.250: icmp_seq=5 ttl=255 time=1.4 ms

--- 192.168.100.250 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 1.4/1.4/1.6 ms
```

〔図17〕ftp（ファイル受信）結果

```
ftp> mget rcv_sample.data
mget rcv_sample.data? y
200 PORT command successful.
150 Opening BINARY mode data connection for
'rcv_sample.data' (2068083 bytes).
226 Transfer complete.
2068083 bytes received in 199.85 secs (10.1 kB/s)
ftp>
```

〔図18〕ftp（ファイル送信）結果

```
ftp> mput snd_sample.data
mput snd_sample.data? y
200 PORT command successful.
150 FILE: snd_sample.data
226 Transfer complete.
2743170 bytes sent in 254.21 secs (10.5 kB/s)
ftp>
```

16）．

（シェルコマンド）

```
$ ping 192.168.100.200
```

● 実用例

PPP接続が完了し，TCP/IPの機能を使用する各種アプリケーション（ftp，telnetなど）が利用可能になりました．ここでは，データの転送レートの検証をかねて「ftp」でファイルの送受信を行ってみます．

送信，受信ともに10Kバイト/s程度の転送レートが出ていることが確認できました．UARTの転送レートは115.2kbpsなので，ほぼ限界値に近い速度でファイルの転送が行えています（**図17**，**図18**）．

3 Bluetooth対応の製品開発について

Bluetooth対応の製品開発を行った経緯から，開発にあたってのポイントをいくつか取り上げます．

At-BT，およびAt-BT-EVAの開発にあたっては「仕様の解釈」，「バージョン互換」，「ベースバンドモジュール入手」など，いろいろな問題がありました．

Bluetoothの仕様は「Core」，「Profile」という2種類，あわせて1700ページにもおよぶ仕様書にはじまり，テスト仕様書，ERRATA，および現在も策定中の新規のプロトコル・プロファイルと多岐にわたっています．その結果，ベースバンドモジュールをはじめとするハードウェア部品，インターフェース部にBluetoothを使用する製品（プリンタ，ワイヤレスモデム），または，At-BTのような実際に製品に組み込むためのソフトウェア（プロトコルスタック）といったように，多種多様な製品が存在します．そのため，前述の内容のほかにもさまざまな問題点があります．

その中の一つで，Bluetooth対応の製品開発を行ううえで共通の課題である「ロゴ認証システム」について解説します．

● ロゴ認証システム

Bluetooth搭載の機器を，正式に「Bluetooth対応機器」として市場に販売するためには，Bluetooth SIGが定める認証基準の審査をクリアし，Bluetoothの仕様に準拠した機器の証明である

「ロゴ認証」を取得しなければなりません．

審査の際の製品開発元の作業手順は，次のようになります．

- Bluetooth SIGより公開されているテスト仕様書(ICS)より，申請を行うテスト項目を抽出する
- 抽出したテスト項目についての動作を証明するテスト結果を作成し，BQTFに報告する

このテストおよび結果の抽出には，多くの労力とコストを要します．At-BTのようなプロトコルスタックでは，ロゴ認証の取得完了までの期間として約3か月かかるといわれています．

At-BT，At-BT-EVAの認証作業を経た上で，次のようなポイントがあげられます．

- Core，Profile全般にわたっての理解が必要

とくにProfile(SDAP，DUNPなど)についての認証取得の際には，Coreにて規定されている各レイヤ(LMP，Baseband，L2CAPなど)全般にわたってテスト項目が準備されています．よって，当然のことながらCore，Profileに関しての幅広い理解が必要になります．

- テスト結果(レポート)の作成

TEST Specificationの記述に忠実にしたがったテスト結果ログの作成を求められます．

At-BTの場合はプロトコルスタックなので，対応レイヤごとにログ出力用のプログラムを作成する作業が発生しました．これだけでかなりの期間を必要としました．「テスト」という名前の示すとおり，あとは合格点が取れるまでひたすらレポートを作成して報告する形になります．

また，このテスト項目の抽出は内容により必須項目(必ずテストを要する)と，任意項目(機能の実装は必須ではない)とに分かれており，任意のテスト項目について申請するかどうかは開発元の判断にまかされているのが現状です．このため一口にロゴ認証取得製品といっても，実装された機能にバラツキが生じてしまいます．

この問題を回避する方法の一つとして「既存のBluetooth対応機器との接続を行い，テスト結果を作成する」というテスト項目が用意されている点があげられます．

また，UnPlug Festa(相互接続性のチェックを行うセミナ)が開催され，実際にほかのBluetooth対応機器との接続を行うことで，相互接続性を高めようとしています．

以上のような状況は，Bluetooth対応機器の信頼性を高める反面，ダイレクトに製品コストに反映されてしまうという問題点でもあります．

おわりに

ここ数年の無線LANの盛り上がりをみると，802.11bをはじめとする無線技術の普及・進歩が大幅に進んだことは明らかです．残念ながら，その仲間の一つにBluetoothが入っているとは現段階ではいえません．

しかし，BluetoothをはじめとするWPAN(Wireless Personal Area Network)の必要性と普及の可能性は，まだまだ未知数です．

そのうえでBluetooth開発に携わった一開発者として，Bluetoothのさらなる普及，そしてAt-BT，At-BT-EVAがその推進の一翼を担えることを願っています[注1]．

参考文献/URL

1) Bluetooth SIG公式サイト　http://www.bluetooth.org/
2) IEEE 802.15 Working Group for WPAN　http://grouper.ieee.org/groups/802/15/index.html
3) PPxP公式サイト　http://www.linet.gr.jp/~manabe/PPxP/
4) Armadillo公式サイト　http://armadillo.atmark-techno.com/
5) 杉浦彰彦，『Bluetooth技術解説』，ソフト・リサーチ・センター
6) 特集「近距離通信技術Bluetoothのすべて」，『Interface』，2001年8月号
7) 特集関連「Armadilloの概要と使用方法」，『Interface』，2002年7月号

なかの・ゆきひと　(株)アットマークテクノ

注1：本章で解説したサンプルプログラムのソースコードは，本号付属CD-ROM「InterGiga No.29」に収録している．

無線LANの基本技術を実装する

Chapter 3

OFDM無線モデムの基礎技術と設計事例

● 西村芳一

無線LANの国際標準規格IEEE802.11aでは，変調方式として，移動体通信に向くといわれるOFDMを採用している．また，地上波ディジタルテレビ放送でもOFDMが採用される．本章では，このOFDMの技術的基礎をまず説明する．そして，OFDMデータモデムの設計事例を紹介しながら，開発のポイントをくわしく解説する．

（編集部）

はじめに

近年，無線LANやディジタル地上波テレビ放送関連で，**OFDM**（**Orthogonal Frequency Division Multiplexing**）と呼ばれる変調方式をたびたび耳にします．本誌でも何度か，その技術的解説がなされてきました．読者の皆さんも関心が強い技術の一つだと思います．しかし，原理的なことは理解できても，実際にその技術を使い商品設計に生かせる具体的な記事は少ないような気がします．ここでは，経験豊富とはいえないかもしれませんが，実際にOFDM無線モデムを商品化設計したとき（**写真1**）の事柄を中心に，OFDMの解説を試みます．

筆者も便利に使わせてもらっていますが，MATLAB（**写真2**）は，このような通信分野での事前のシステム評価には欠かせないものとなっています．以前は，それぞれのシステムにあわせて**図1**[注1]のようなシミュレーションソフトを自前で作らなければならず，時間と手間がかかったものです．しかし，そうして苦労して得られる結果は，商品化設計になくてはならないものです．実際にハードウェアを作り上げる前にある程度，さまざまな条件でテストが可能で，苦労してハードを作ったあと，使いものにならないといったような無駄な状況を回避できるようになってきています．

MATLABが出力する結果を使い，手軽に細かな評価もできるようになった反面，それですべてを理解したような気分になります．しかし，それから商品化設計が始まるのです．実際，そこには，理論をいかに具体化するかの難しい問題と，シミュレーションでカバーできなかった実動作の厳しい評価が待ち受けているのです．

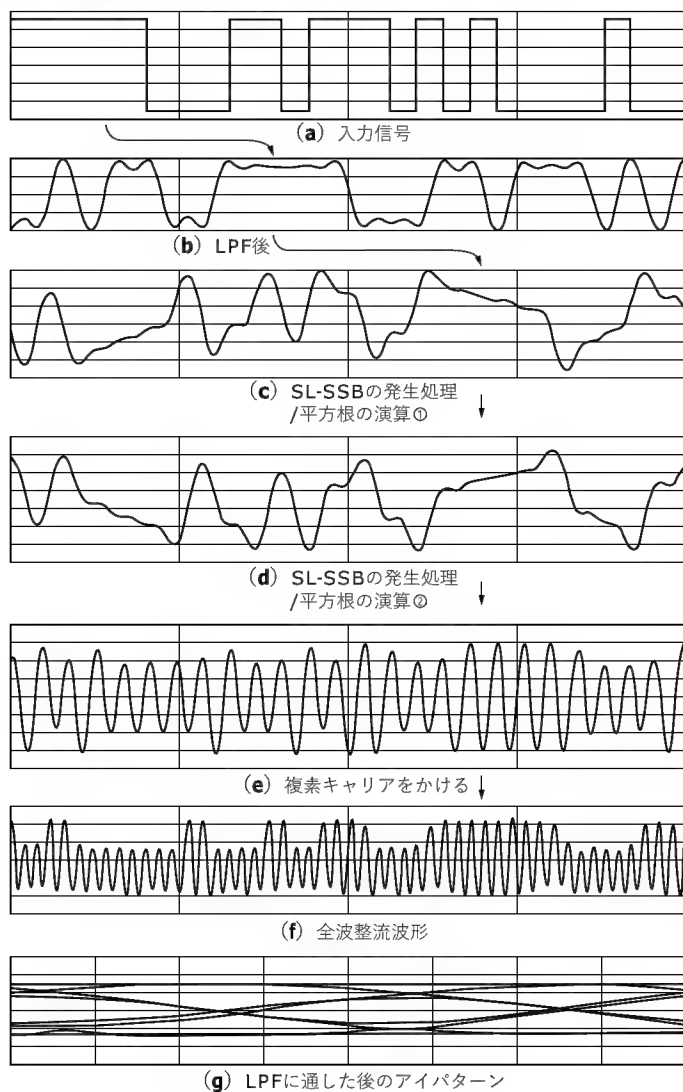
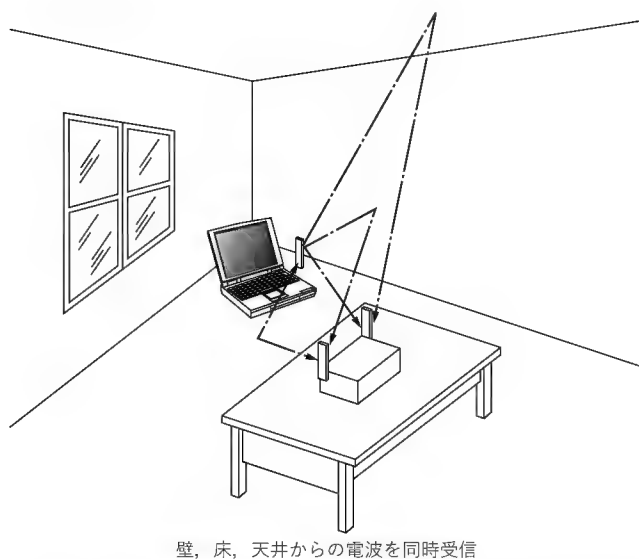
1 無線によるOFDMの基礎

1.1 なぜ今OFDMか

無線特有の問題として，マルチパスやドップラ効果などがあり，これらを避けることはできません．マルチパスでいえば，と

注1：SL-SSB（Square-LanSSB）．自乗検波で，復調が可能な変調方式．くわしくは参考文献7）を参照．

〔写真1〕OFDM無線モデム

〔写真2〕MATLAB

くにLANなどの場合は状況が深刻です．室内で使われることが多いと思いますが，**図2**のように壁，床，天井などからさまざまな反射波を同時に受信することを覚悟しなければなりません．これは，重大なフェージングの問題を引き起こします．

また，最近のデータ伝送はどんどん高速化の方向に向かっており，データ伝送における1シンボルの時間間隔はどんどん短くなる一方です．これにより，**図3**のようにちょっとした遅延のあるマルチパスの信号でも重大な影響を与えることは簡単に予想できます．従来のシングルキャリアの高速モデムでは，部屋の中の受信場所をちょっと変えるだけで，受信不可能な状態に陥ることが考えられるのです．

そこで，マルチパスに強い，OFDMに注目が集まるようになりました．**図4**のようにキャリアの数を増やせば増やすほど，シンボルレートを遅くすることが可能です．すなわちシンボル同士の時間間隔を広く取れ，シンボル間の干渉の影響を受けにくい，マルチパスに強いシステムが構築できます．さらに後で説明するガードインターバルを挿入することにより，より組織的にマルチパスからの影響を軽減することが可能となっています．

固定局同士の無線通信の場合は，ある程度指向性の強いアンテナを使うことによりマルチパスの影響を少なくできます．しかし，無線LANのように部屋のどこで使うかわからない移動体無線の場合は，アンテナに指向性をもたせることはできません．そのため，マルチパスによるフェージングを避けることは難しく，OFDMに注目が集まっているのです．

OFDMが優れた変調方法の一つであることは皆さんすぐに理解できるでしょう．一方で，それぞれのキャリアにPSKなど複雑な変調をかけなければならず，とても複雑で難しい技術のような印象を与えているのも事実です．しかし，OFDMとフーリエ直交変換を結びつけることで，誰でも手の届く技術となっているのです．フーリエ変換については，皆さんはすでによく知っていると思います．信号処理の基本となるもので，**図5**に示すように時間軸の信号を周波数軸に変換して信号を解析利用するものです．“F特”といえばふだん仕事で必ず出てくる言葉で

〔図1〕SL-SSBシミュレーションの例

(a) 入力信号

(b) LPF後

(c) SL-SSBの発生処理/平方根の演算①

(d) SL-SSBの発生処理/平方根の演算②

(e) 複素キャリアをかける

(f) 全波整流波形

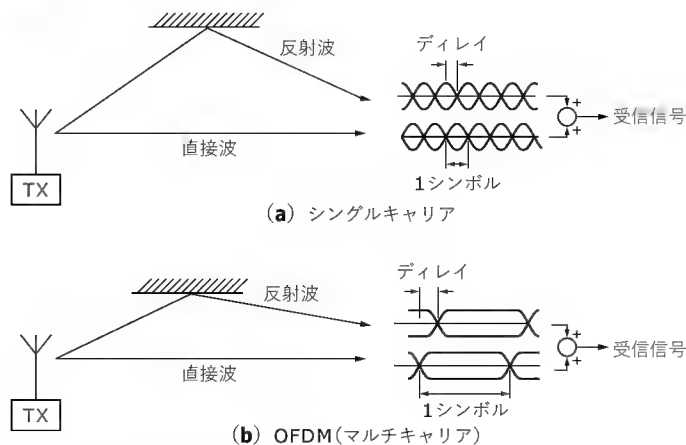
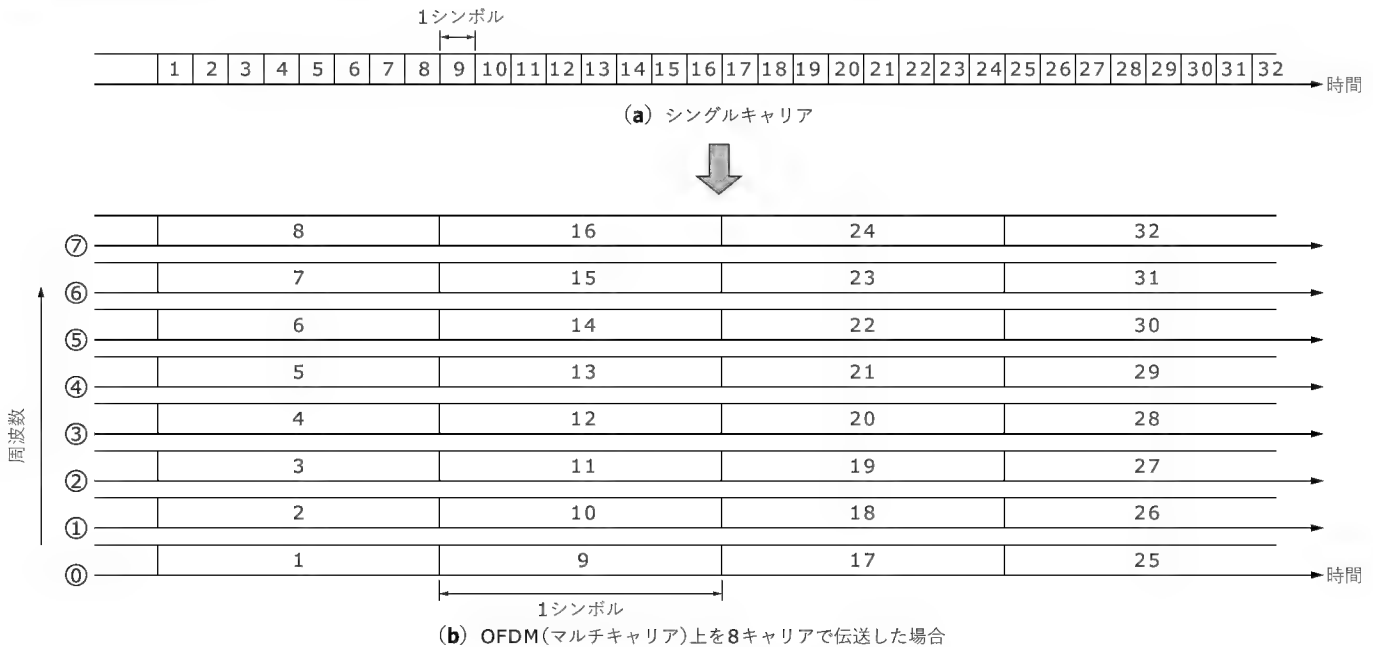
(g) LPFに通した後のアイパターン

〔図2〕マルチパス

壁，床，天井からの電波を同時受信

〔図3〕マルチパスの影響

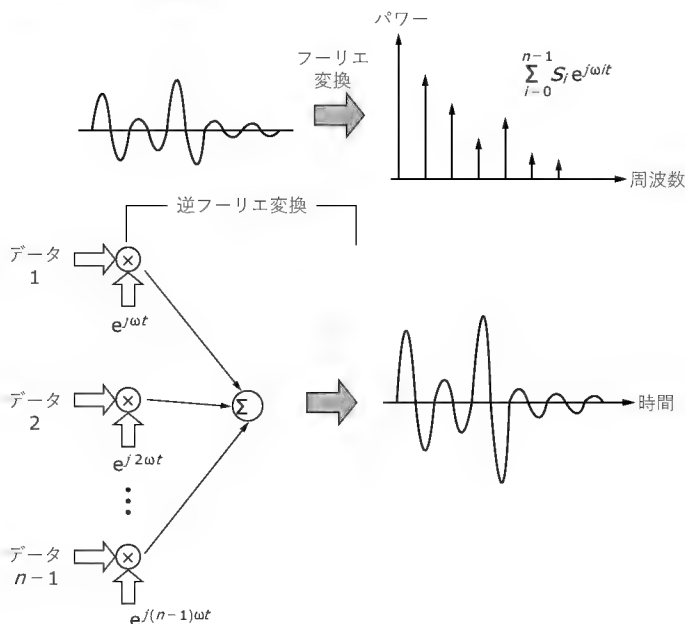
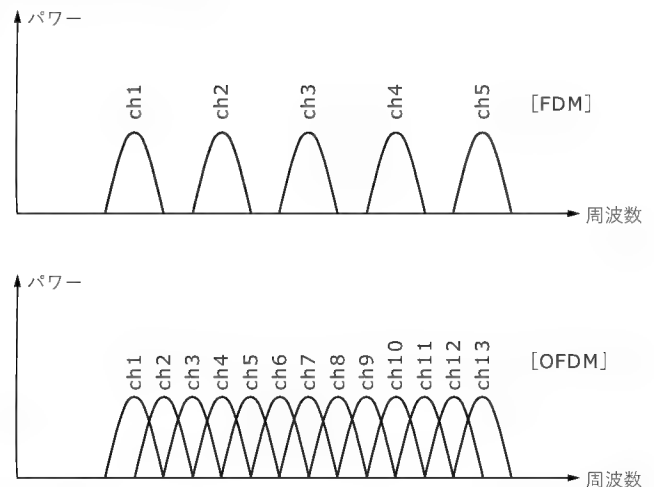
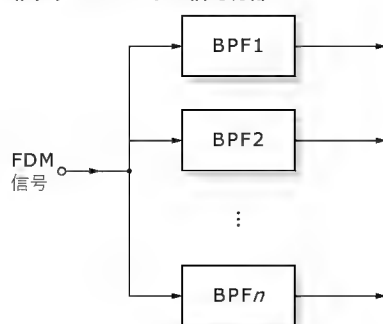
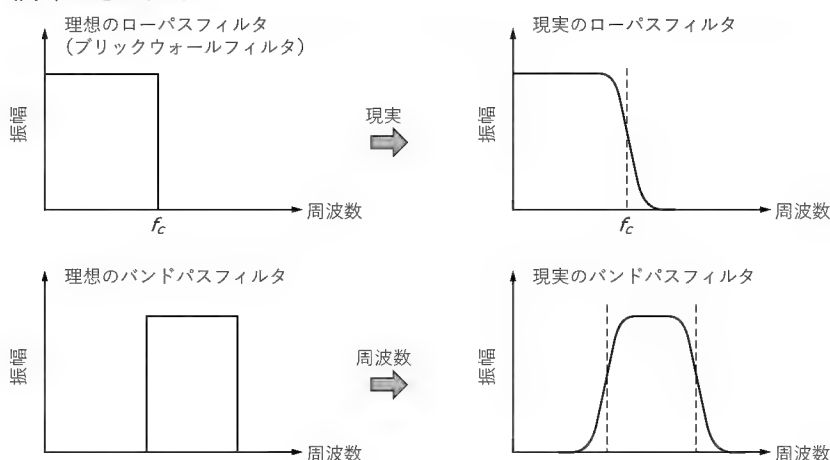
(a) シングルキャリア

(b) OFDM(マルチキャリア)

〔図4〕低速シンボルレート

す．ふだん何気なく，いろいろな性能の評価に周波数軸での解析を行っています．この図を見れば，何となくOFDMのようすが予想できるのではないでしょうか．

1.2 歴史

OFDMの基本的な考え方自体は，けっこう古いものです．直交性という条件をはずせば，基本的には**図6**のように周波数で信号を多重し，効率よく伝送する考え方です．無線，もっと身近にいえばラジオでわかるように，ある放送を聞くためにはチューニングをして目的の放送局の周波数にあわせます．もし手元に複数のラジオがあり，それぞれ別々の局のチューニングをすれば，同時にいろいろな番組が聞けます．実際の問題は，聖徳太子でもないかぎり同時に多くの情報を聞くことはできないので，ただうるさいだけでまったく意味がないことです．しかし伝わる情報量がチャネルの数に比例することはおわかりでしょう．

従来の周波数多重(FDM ： Frequency Division Multiplexing)では**図7**のように複数の低速で変調されたキャリアを，それぞれ別々のBPF(バンドパスフィルタ)で信号を分離するしかありません．BPFが**図8**のような理想的なものがあれば(ブリックウォールフィルタと呼ばれる)，隙間なくチャネルをつめることができます．実際には不可能なので，チャネル間には信号の伝送に直接関係のない(周波数的に無駄な)ガードバンドが必要になります．

〔図5〕フーリエ変換

〔図6〕FDMとOFDM

〔図7〕FDMからの信号分離

〔図8〕理想フィルタ

まず1950年頃，このガード周波数バンド付きのマルチキャリアを使ってパラレルデータ伝送する方式がアメリカで特許化されています．しかし，先ほど述べたようにガード周波数バンドの分，周波数利用効率が悪いものでした．

そこで1960年代の中ごろになると，キャリアの周波数間隔と変調帯域が同じような効率的な方法が提案されました．見かけはOFDMにほぼ近いものの，キャリア同上は直交していません．そのため，キャリアのサイドローブ同上が重なり，それを取り除くために高速な等価器が考えられました．ただし，等価器だけを使って，それぞれの信号の干渉を完全に取り除くことはとても難しい技術でした．

ここでOFDMが登場します．マルチキャリア同上の相互相関をゼロ，すなわち**図6**のような直交キャリアを使い，サイドローブが重なっても，それぞれの信号干渉をなくすように考えられました．変調される信号がアナログの場合では，キャリアの両側に帯域が広がって直交条件を満足するのは難しいことです．サンプリングされたディジタル信号を変調に使うことにより可能となったものです．理論上はとてもスマートな変調方式ですが，実際のリアルタイム信号処理としての応用には高い壁がありました．

〔図9〕DRM

そのような状況の中でブレークスルーをもたらしたのは，1971年にWeinsteinとEbertによって提案された，**図5**のような離散フーリエ変換をOFDMの変調復調に使う方法です．これまでハードウェアのリアルタイム処理では不可能と考えられていた処理が，がんばれば実現できそうなところまで降りてきたのです．もともとフーリエ変換は直交した複素正弦波を使うもので，それぞれのキャリアのQPSKなどの複素変調と，とても相性がよいものです．

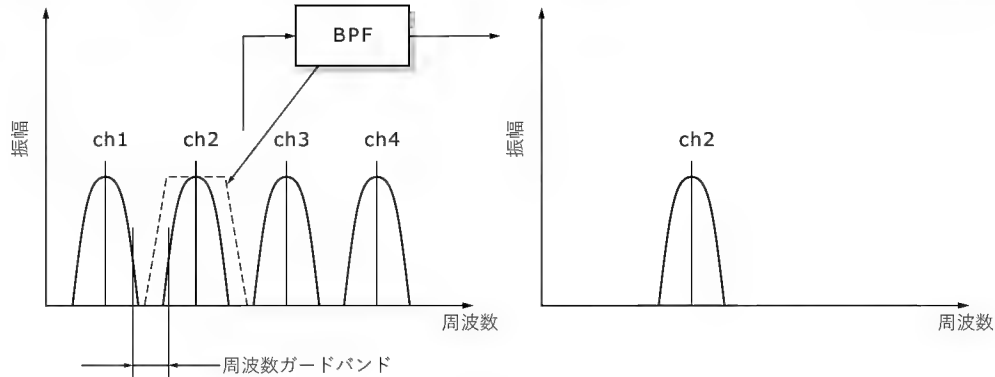
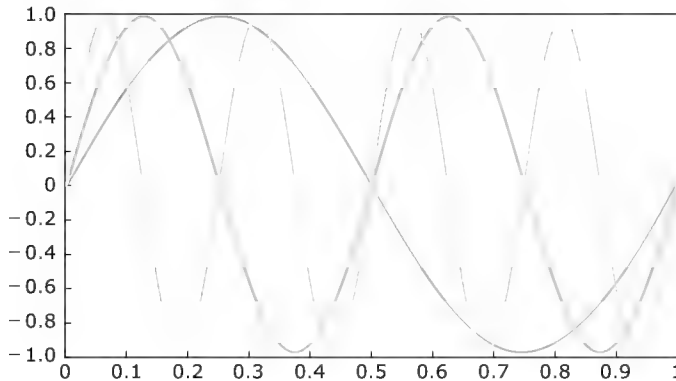
初めてのOFDMの無線への応用は，1960年代軍用に作られたHFトランシーバです．キャリア間隔が82Hzで，34のマルチキャリアを使うものです．その後1980年代に入り，移動体無線に向けて高速データ通信が研究されました．それからデータ伝送速度を速くするため，変調もより複雑になり，それぞれのキャリアはQAMで変調されトレリスコーディングが用いられるようになりました．

最近は地上波テレビジョンのディジタル化など，放送におけるOFDMの採用が進んでいます(実際には符号化にコンプリメンタリコーディングを用いているのでCOFDMと記述されている)．その裏には半導体技術の高集積化高性能化が見逃せません．OFDMの受信機を将来の量産時に安く作る見込みがなければ，OFDMは採用されなかったでしょう．その基本は，専用LSIの開発いかんにかかっています．

さらに**図9**のように，いよいよ日本を含め世界中の放送局などがメンバに入った組織DRM(Digital Radio Mondiale)により推進される中波，短波のディジタル化が現実のものとなろうとしています．電波伝搬は電離層反射によるもっとも過酷なフェージングの中を伝送する必要があり，これも当然のことながらOFDMを使っています．高品位のFM放送や衛星放送があるの

〔図10〕FDMの分離

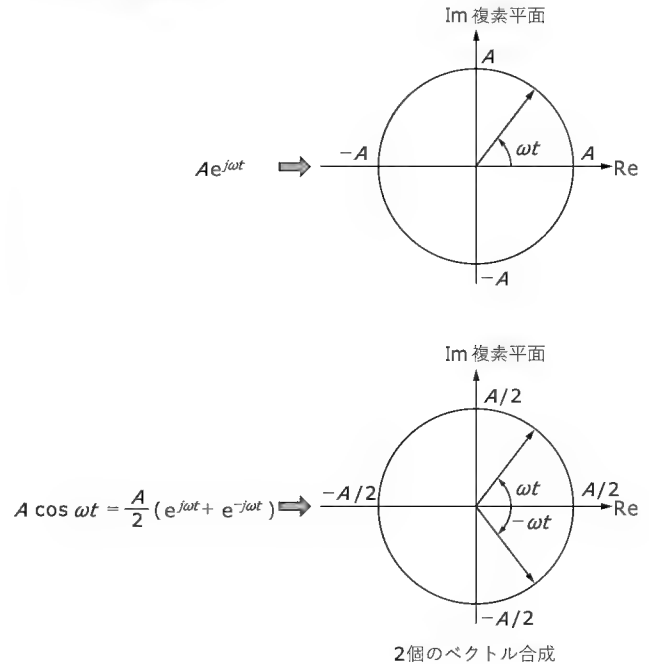
〔図11〕フーリエ変換のsin基底

〔図12〕複素正弦波

に，いまさら中波のディジタル化かと思われる方も多いでしょう．筆者も詳しいところはわかりませんが，ドイツの関係者に話を聞くと，国内のFM放送のカバー率が低いため依然中波の高品位放送の要求があるそうです．2003年から，本格的な実験放送が世界的に始まる予定です．それに向けたラジオ用のLSIの開発が各社いっせいに進められています．FM放送なみの高品位ステレオ放送をめざしています．

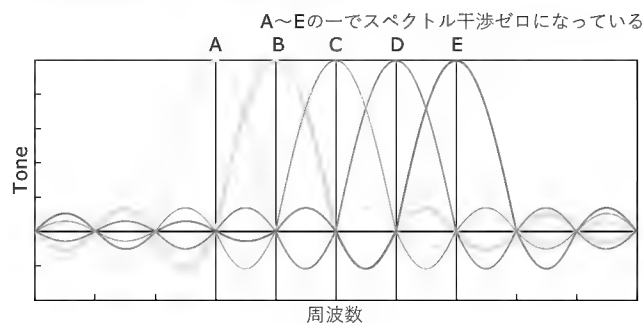
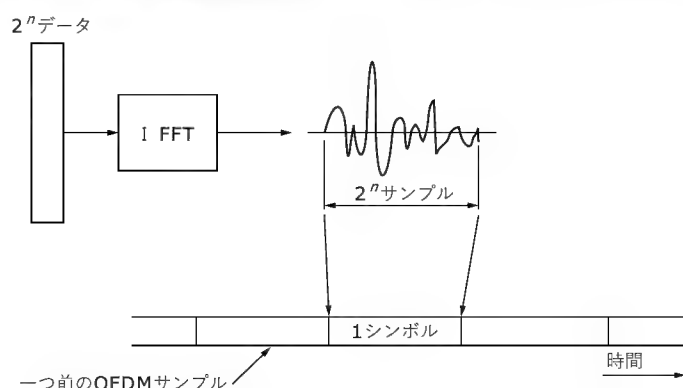
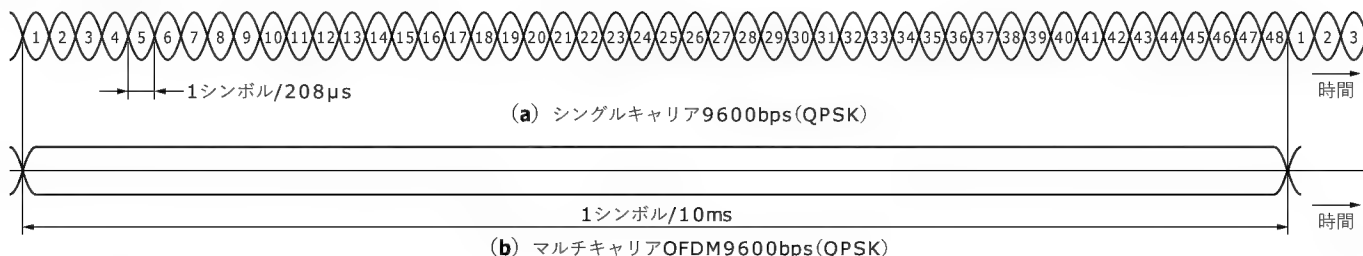
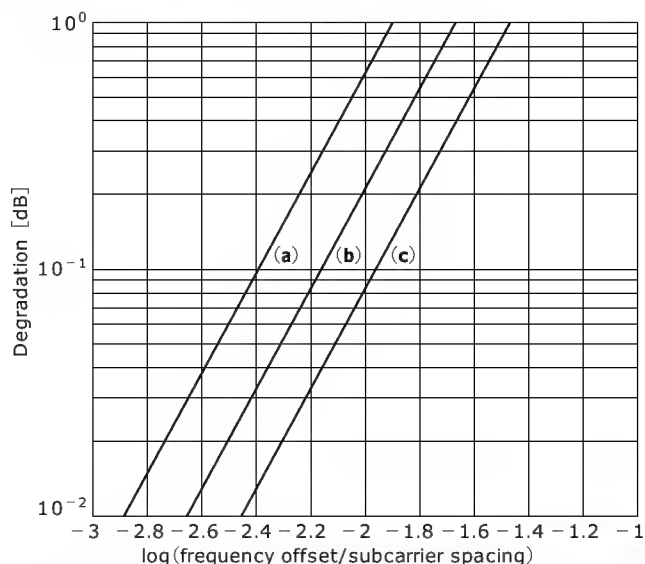
マルチパスによる影響が大きい無線LANでは，たとえば**Bluetooth**がホットな話題です．Bluetoothでは，これを**周波数ホッピング**を使うことにより対処しています．周波数拡散による方法も有力な手段ですが，とくに最近5GHz帯のIEEE802.11を代表とする無線ブロードバンド通信の分野では，多くがOFDMで標準化されてきています．有線系でも，ADSLや電力線を利用したブロードバンド通信においてもOFDMの変調が使われています．

1.3 直交キャリア

周波数多重した信号は，受信側で元に戻す必要があります．従来のフィルタによる方法だと，**図10**のようにフィルタのスカート部分をカバーするようなガードバンドがどうしても必要です．OFDMでは**図6**のように極限までキャリアを詰め込んでいます．これを分離するためには，信号の直交性を利用しなければなりません．

関数同士が直交関係にあるとき，相互相関はゼロにならなければなりません．すなわちコンボリューションがゼロになります．そのような関数群はいろいろな種類がありますが，OFDMで使うのは，フーリエ変換で使う直交関数である，複素正弦波になります．**図11**に，フーリエ変換で使う三角関数の基底の一部を実時間波形で示します．OFDMでの応用では，すなわちそれぞれの複素正弦波を複素変調してPSKやQAMを作り出します．

複素正弦波といってもなかなかピンとこないかもしれません．**図12**のように複素平面上の円周上を一定速度で回転するベクトルをイメージすれば理解しやすいと思います．しかし複素数は仮想的信号なので，**図12**のようにこのベクトルと反対方向に回転する共役ベクトルとの和を考えると，現実の実数のみの信号が得られます．その信号が実際観測できる信号波形であると考えるものです．

直交キャリアのイメージをわかりやすくするため，各キャリアのスペクトルを重ね合わせたものを**図13**に示します．キャリア間のスペクトル的な干渉は，各キャリア周波数のところで干

〔図13〕直交キャリアのスペクトル

〔図14〕OFDMの繰り返し構造

〔図15〕9600bpsシングルキャリアvsOFDM

〔図16〕周波数オフセットと特性劣化

(a) 64-QAM(E_s/N_o=19dB)
(b) 64-QAM(E_s/N_o=14.5dB)
(c) QPSK(E_s/N_o=10.5dB)

Richard Van Nee,
Ramjee Prasad,
OFDM FOR Wireless Multimedia Communications,
Artech House, 2000

渉がちょうどゼロになっていることがよくわかります．

したがって，信号の分離は受信した信号と各複素正弦波とのコンボリューションをとることになります．すなわちフーリエ変換すればいいことになります．

ここで注意したいのは，このような基底を使っても各キャリアを従来のようにアナログ変調したのでは直交関係は崩れるという点です．**図13**のように干渉がないのは各キャリア周波数1点のみです．アナログ変調では連続関数なため，変調した後の信号帯域が広がって干渉が発生してしまいます．OFDMでは，フーリエ変換の区間は変化しない一定のサンプルされたデータを使うので，キャリア帯域は広がらず直交関係を保てるのです．

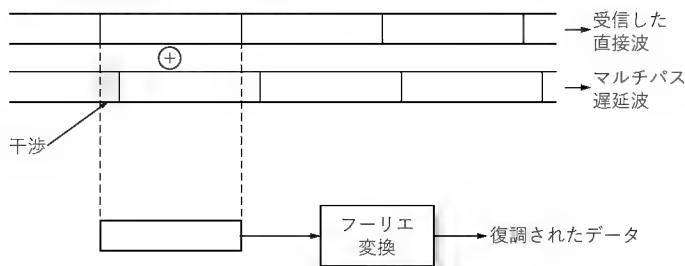
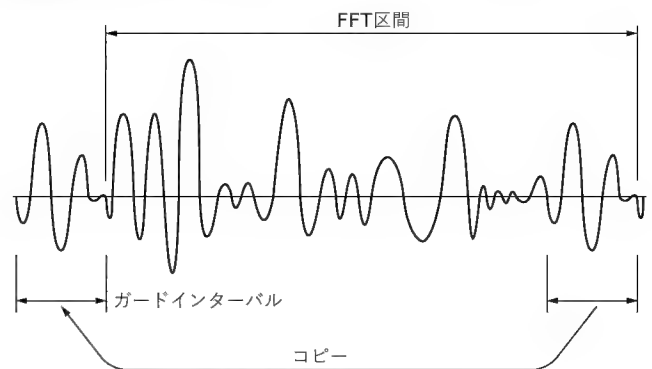
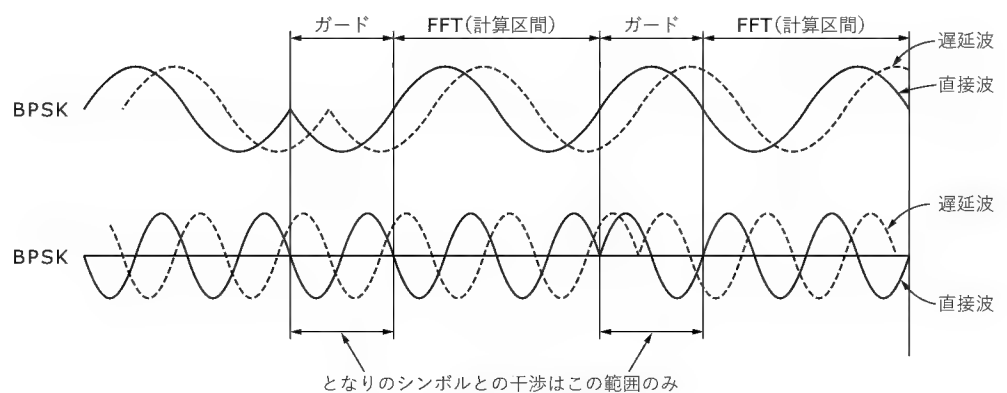
1.4 信号の変調

離散的フーリエ変換を計算するとき，実際には有限区間の高速フーリエ変換(以下FFT)を使います．そのため，FFTの基本周波数の区間で計算を行います．すなわち**図14**のような繰り返しの信号構造になります．この繰り返しの1区間をOFDMの1データシンボルとしています．

各キャリアをQPSKで変調したとすると，1データシンボルで1キャリアあたり2ビットを伝送することが可能です．すなわち全体ではキャリアの数をNとし，1シンボル区間をT(秒)とすると，$2N/T$(bps)のデータ伝送速度を得ることができます．

ここで具体的な例として9,600bpsのデータ変調を考えてみます．従来のシングルキャリアのQPSK変調を考えると，**図15**のように1シンボルは約208μsとなります．ところが48本のキャリアを使ってそれぞれQPSK変調を行ったOFDMでは，1シンボルが10msとなります．OFDMでは，シンボル間隔を長く取ることが可能で，マルチパスに強いことが容易にわかると思います．

もちろん良いことばかりではありません．OFDMでは時間軸のひずみについては強いのですが，その反面，周波数軸のひずみについては弱点があります．マルチキャリアにすることによ

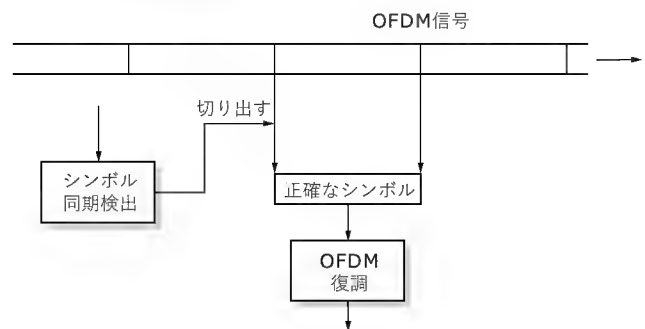
〔図17〕受信側ブロック処理

〔図18〕ガードインターバル

〔図19〕各キャリアBPSKのときの
ガードインターバル

り，各キャリアの周波数配置間隔がとても狭くなります．すなわち，ドップラーシフトや同調ずれなどで周波数のオフセットが発生した場合にはキャリアの直交関係が崩れ，相互干渉が発生してしまいます．**図16**にその影響をまとめてあります．キャリア周波数間隔の半分以上の周波数オフセットが発生すると，ほとんど使いものになりません．

1.5 ガードインターバル

先ほど，1シンボルの時間が長くなるのでマルチパスに強いと書きましたが，**図17**のように受信側ではシンボルごとにブロック処理がなされるので，実際にはマルチパスが発生すると，隣のシンボルと干渉を起こします．

そこで，**図18**のようにマルチパスの最大時間を見越した分だけシンボルの前にダミー信号を入れます．具体的にはシンボル波形の後半部をそのままコピーして貼り付けます．ただし実際の復調に際しては，ダミー部分は含めないで行われます．すなわち**図19**のように，マルチパスにより遅延してきた前のシンボルの後半部との境目は，そのダミー信号の領域に収まります．つまり，このダミー信号区間の信号を復調の計算には使わないようにすることにより，シンボル間の干渉を防ぐことができます．

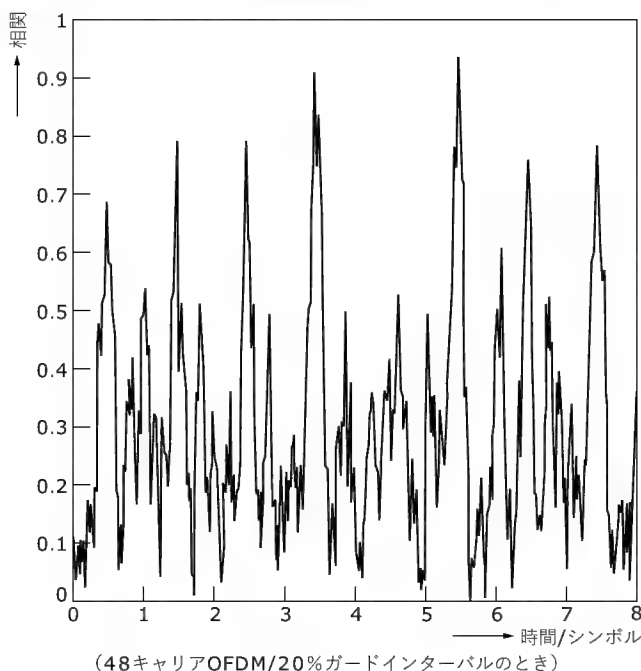
このダミー信号はシンボル干渉をガードするという意味で，ガードインターバルと呼ばれています．ガードインターバルは，信号伝送には直接関係ないので，データ効率を考えると無駄です．しかし，OFDMの変調においてはなくてはならないものです．

さらにこのガードインターバルには，もう一つ大きな役割があります．OFDMでは，1シンボルごとに復調が行われます．となり同士のシンボルはガードインターバルを挟んだだけでつながっています．受信側で復調の際にシンボル干渉が発生しないのは，あらかじめシンボルの区切りがわかっている条件のときのみです．そこで**図20**のように，受信側においてシンボルの区切りがわかるようなシンボル区間の検出と同期の機能が必要です．

〔図20〕シンボル同期

しかし，シンボル同期を検出するような特別の信号は，OFDMの信号の中に含まれないのが一般的です．そこで，このガードインターバルの信号の性質を利用します．**図19**のようにガードインターバルとシンボルの後半の波形はまったく同じ波形をしています．その間の相関を計算すると**図21**のように鋭いピークをもつ値を検出することが可能です．これを検出することによりシンボル同期をかけることができるのです．しかしまた，この

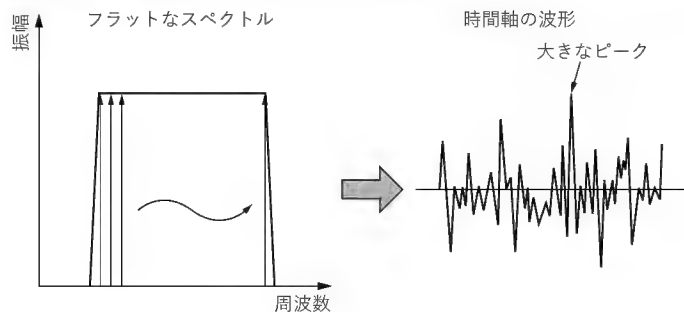
〔図21〕ガードインターバルとの相関

（48キャリアOFDM/20%ガードインターバルのとき）

〔図23〕OFDMのスペクトル

図はピークを見つけることの難しさも示しています．絶対値でスライスしてそれを超えたものをピークとするようなことはできません．また，何らかの同期処理が必要となります．

ガードインターバルを超えるようなマルチパスが発生した場合は，干渉を起こします．それがどの程度なのかを**図22**に示します．3%のオーバラップでも，かなりコンスタレーションが乱れています．ここでは16QAMで変調しているので，かなりクリティカルなのがわかります．

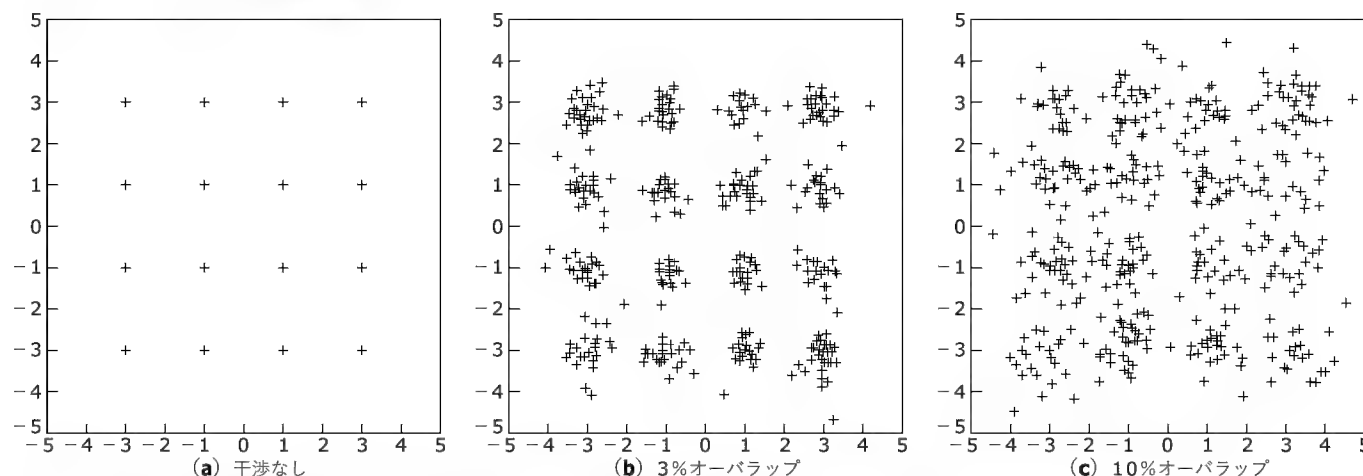
1.6　ダイナミックレンジの問題

従来のシングルキャリア変調に比べて利点が多いOFDMですが，前に述べた周波数オフセットに対して問題を起こしやすいのに加え，もう一つ大きな問題があります．携帯電話をはじめとする移動体無線機はバッテリで運用されます．その場合，1回の充電でどれくらい使えるかの電池寿命は，設計のなかで重要な要素です．中でも電力を多く消費する送信側の終段アンプは，とくに気を使って設計する必要があります．

FM変調系などの非線形変調は，変調された信号の振幅が一定です．そのため，出力アンプとして電力効率の良いC級アンプが使えます．一方QPSKやQAMなどの線形変調では，出力アンプも線形アンプを使って歪みを極力避ける必要があり，効率の悪いA級もしくはAB級アンプを使う必要があります．

そのため，多くのディジタル無線機では，現在でも変調として非線形変調のGMSKが使われています．たとえば，欧州や中国など世界の多くの国で使われているGSM携帯電話は，GMSKが使われています．

線形変調の仲間のOFDMは，QPSKなどよりさらに深刻な問題があります．**図23**のスペクトルからわかるように，変調信号は広い周波数範囲にまたがってフラットな周波数特性を示します．言い換えれば，ホワイトノイズと同じようなランダムな波形をしています．そのため，図のように信号のピークと平均値の比がとても大きくなります．

そのような大きなピーク波形を歪ませないようなアンプを設計すると，必要以上に大きな出力増幅器をつくらなければなり

〔図22〕マルチパスによる干渉の影響

(a) 干渉なし　(b) 3%オーバラップ　(c) 10%オーバラップ

Richard Van Nee, Ramjee Prasad, *OFDM FOR Wireless Multimedia Communications* Artech House, 2000

〔図24〕同期ヘッダ(プリアンブル)

ません．当然A，AB級の場合，それだけ無駄なアイドリング電流を流さなければならず，とても効率の悪いものになってしまいます．かといって小さいアンプを使ってピークを歪ませないように出力を絞ると，伝送上必要な平均値のパワーがとても小さくなって使いものになりません．

これを対策するためには，いくつかの方法があります．一つは符号化の部分でピーク・平均値の比を考慮に入れた設計をする方法，もう一つは何らかの方法でピーク波形を制限する方法です．くわしくは，後ほど説明します．

1.7 シンボル同期

先ほどガードインターバルのところでも記したように，受信側でシンボル間の干渉もなく正しく復調するためには，シンボル同期が必要です．シンボル同期を取るためには時間軸の同期と，周波数軸の同期の二つを考える必要があります．またもっと広い範囲で考えれば，伝送で発生した特性の劣化，歪みを自動補償することも含まれます．

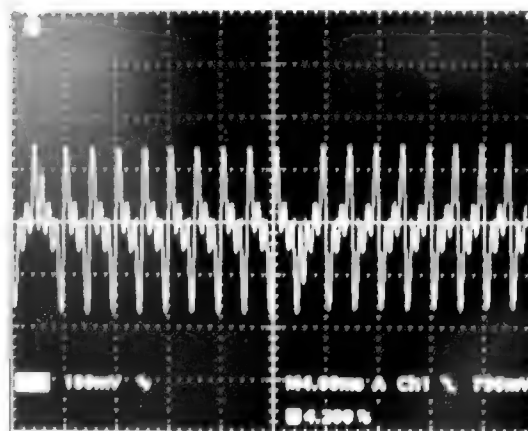
1）連続送信とパケット送信

同期や，さまざまの特性の自動補償を考えるとき，OFDMのデータの種類を考える必要があります．一つは地上波ディジタルテレビなど送信信号が常に出力されっぱなしで，受信側はいつ受信状態に入るかわからないデータ伝送の状況です．この場合は，いつも送られているデータ構造の中から同期に関わる情報が常に得られる状況でなければなりません．

もう一つは，無線LANのようにデータに始まりと終わりがあり，データを一つの集まりのパケットとして送る場合です．その場合は，**図24**のようにパケット構造の中で部分的かつ集中的に同期や補償のためのデータを送ることができます．また特性の自動補償や同期維持に関していえば，パケットデータは短時間で送られます．パケット伝送の最中に伝送回線の特性が急に変化することは考えにくく，放送の場合とは違って，普通のデータの中に参照信号を入れる必要性は少なくなります．

2）プリアンブルを使った時間軸の同期

まずは，時間軸の同期です．それを実現するために，大きく分けて二つの方法があります．

まずその一つに，データを1塊のパケットとして送る場合，先頭に同期を確実に取るような**図24**に示す特殊なヘッダ信号(プリアンブル)をつける場合があります．ランダムなデータではなく決まりきった同期や補償のために特化した最適なパターンの信号を送ることができるので，高速に同期をかけることが可能です．また無線特有の受信機のAGCなどの安定化時間確保のためにも重要な役割を果たします．

〔写真3〕BPSKを使った場合

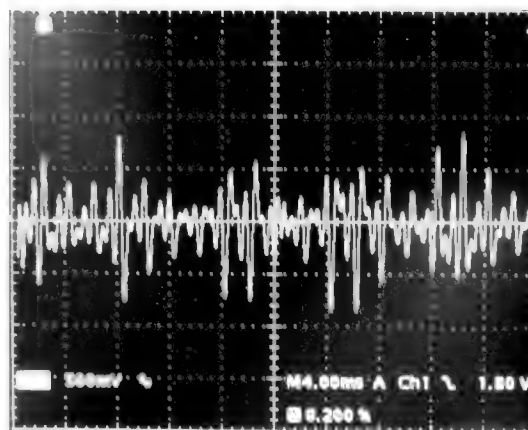
〔写真4〕周波数オフセットが大きい場合

一つの例として，3トーンのキャリアだけを使い，1シンボルごとに位相反転するBPSKを使った場合を示します(**写真3**)．ここでの例は36トーンのOFDMの場合のヘッダ信号ですが，ヘッダのトーン数を制限することにより，一つのトーンのパワーを大きくすることが可能です．つまりフーリエ変換をかけた後の各スペクトルのパワーが大きくとれ，同期がかけやすくなります．

写真3のように，1シンボルごとに位相が180°反転するだけなので，シンボルの区切りがとても見つけやすいのです．この検出には二つの方法があります．一つ目は，先ほどのガードインターバルで説明したとおり，ガードインターバルの波形と信号後半部の波形の相関を計算し，そのピークを探し出して同期がかかったものとする方法です．ランダムな信号と異なり，毎シンボルごとに同じ波形なので，信号の違いによる影響を受けにくくなります．

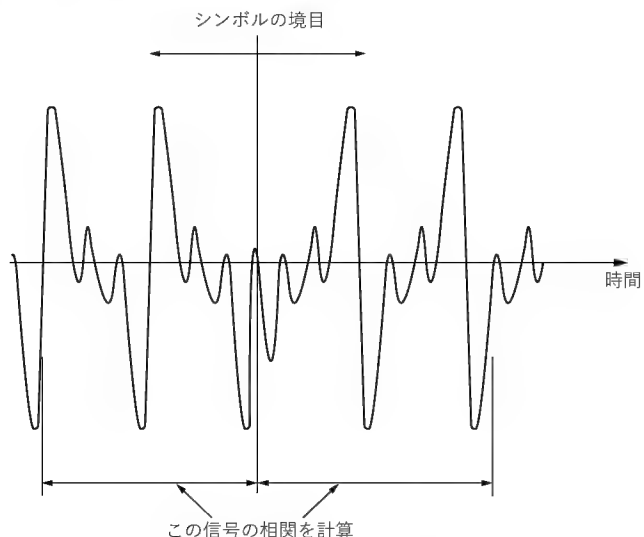
3）シンボルのつなぎ目の性質を使った同期

さらにもう一つの方法として，大きな周波数オフセットをともなう信号の場合は，これより効果的な方法があります．前の相関を検出する方法の場合，**写真4**のように周波数オフセット

が大きいと位相回転により，ガードインターバルの信号と後半の信号の波形が違った形になってしまいます．**写真4**は，**写真3**の波形を約100Hzのオフセットをつけて実際に受信した波形です．まったく異なる波形となっています．

そこでもっと時間差の小さい信号を使います．**図25**のようにガードインターバルとその前のシンボルの後半部の信号との相関を計算する方法が使えます．BPSKで位相が反転していますが同じ波形になるはずです．ただし，この方法は先ほどの方法とは異なり，同じパターンの信号を送るプリアンブルにしか通用しません．ランダムなデータを送る通常のシンボルの場合，前のシンボルとの間に相関はほとんどないからです．

4）ガードインターバルの相関を利用した同期

いったん同期がかかった後のパケットデータの同期維持や，ディジタル放送などの連続データの同期に関しては，**図18**のようにガードインターバルの性質を利用した方法で行います．ガードインターバルの信号は，はじめに説明したようにシンボルの後半部をそのままコピーしています．したがってそれらの間には強い相関があります．その相関を計算した例を**図21**に示しました．

同期が検出できたときには，はっきりとしたピークが現れ，それを検出することにより同期をかけることができます．しかし，信号処理においてピークを正確に検出するのはけっこう難しい技術です．信号の絶対値が絡んでくるからです．

つまり，ピークの大きさは，信号の*S/N*比などに左右されるので，それを考慮した検出が必要です．つまり，受信した信号の*S/N*比やその他信号の品位などを検出し，その状況に合わせたアダプティブな信号処理が要求されます．

5）小さい周波数オフセットの同期

ドップラーシフトやSSBの通信など，送信側との周波数ずれが発生する場合が考えられます．そのズレはキャリア間の直交性を崩し，深刻な干渉をもたらします．周波数オフセットがキャリア周波数間隔の半分以下であれば，次の方法が取れます．特別の信号は使いません．ガードインターバルを使います．

図26に処理ブロックを示します．先ほどの時間軸の相関計算を複素数に拡張します．まず入力信号は，ヒルベルト変換などで複素数に変換されます．または，はじめからIF（中間周波数）から直交復調して*I/Q*すなわち複素数を得ることもできます．シンボルの後半部の信号の複素共役をとります．次式に示すように，その複素共役信号とガードインターバルとの畳み込み積分を計算します．計算した結果もまた複素数になります．

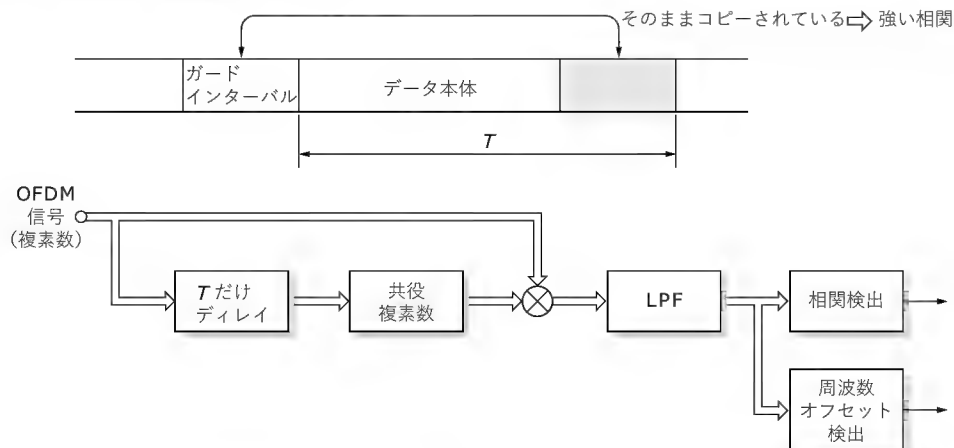
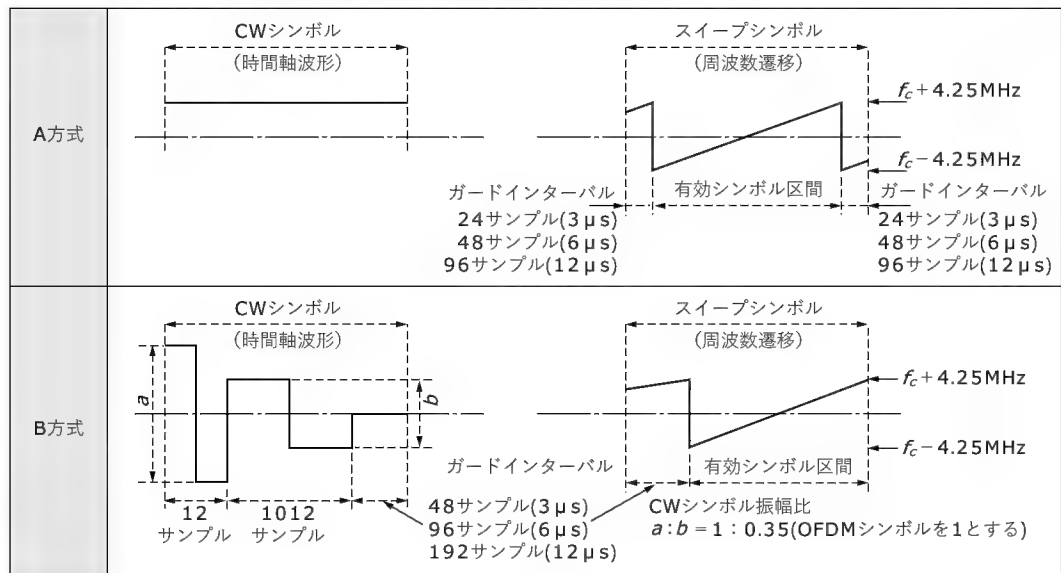
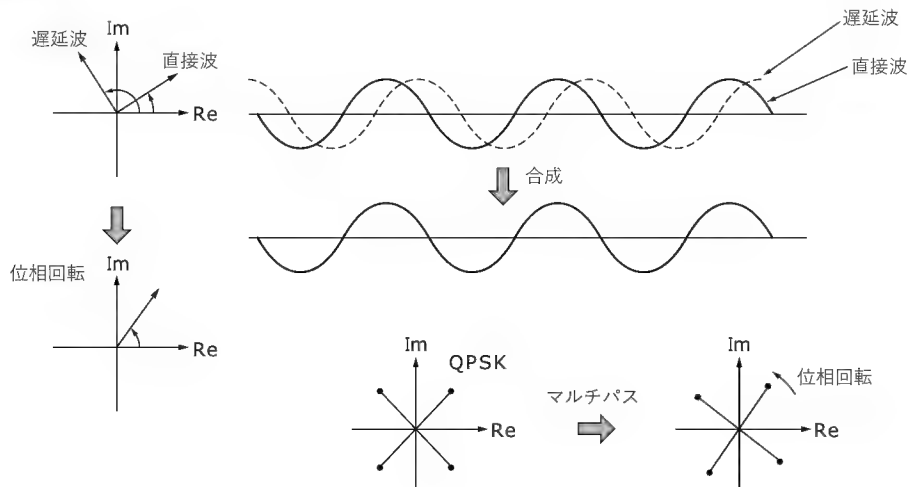
もし周波数オフセットがゼロの場合は，計算結果の虚数成分がゼロになります．周波数オフセットがある場合は虚数成分が残ります．これがゼロになるように補正をかけて，自動的に周波数オフセットの補正が可能です．周波数オフセットのプラスマイナスで，虚数成分の極性も変化します．具体的な周波数補正のやりかたは後で説明します．

6）大きい周波数オフセットの同期

周波数オフセットが大きい場合は，ガードインターバルによる補正はうまく機能しません．その場合は，特別の補償に使う信号を付加します．時間軸の同期のところでも説明しましたが，

〔図25〕シンボル結合部の相関

〔図26〕ガードインターバルを利用した周波数オフセット検出

〔図27〕CW/スイープシンボル信号波形

〔図28〕マルチパスによる位相歪み

データがパケット構造のときパケットの先頭にヘッダ信号がつきます．

たとえば，**写真3**のような3トーンヘッダを使い，周波数の補正を行います．写真のような3トーンヘッダ信号をフーリエ変換を計算すると，周波数オフセットがない場合は正確に1kHz，1.5kHz，2kHzのスペクトルが計算できるはずです．ところが周波数オフセットがあるとそれに比例して，周波数スペクトルのピークがずれるはずです．そのずれ量がゼロになるように自動的に補正をかけます．

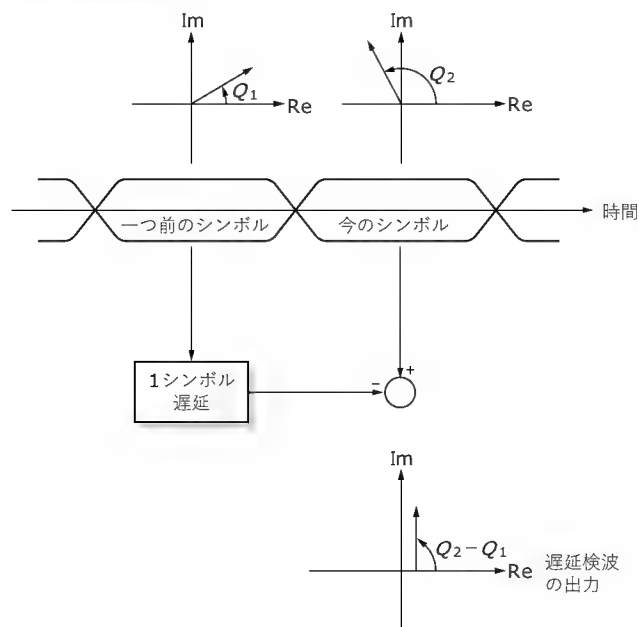
3トーンのヘッダは一つの例にすぎません．その他さまざまな形式が考えられています．たとえば，800MHz帯のテレビ中継用のOFDM規格　ARIB　STD-B13では，**図27**のようなヘッダ信号を使っています．3トーンの変わりに“CW”としてシングルトーンが使われています．またその後には周波数スイープ信号が挿入されています．変調としてはDQPSKを使っていますし，その他の基本的な部分は今回のモデムと大きな違いはありません．

1.8　マルチパスの影響

ガードインターバルがあればマルチパスの影響はないと説明しましたが，これは正確ではありません．**図19**のようにたしかに，隣同士のシンボルの干渉は防ぐことができます．しかし，同シンボル内でマルチパスのため遅延し，位相が異なった信号と直接波との合成した信号による歪みが発生します．

その結果，PSKの場合は，**図28**のように元の位相からマルチパスの信号が加算された分だけ位相回転した信号が得られます．すなわち位相歪みが発生し，位相変調を行うモデムにとっては大きな問題です．PSKの復調の際には，キャリア基準位相再生のときに，この分を自動的に補償しなければなりません．そのために，信号の中にどれくらい位相回転が発生しているのかを調べるためのパイロット信号を埋め込む必要があります．

または，位相の絶対値を使わないで，S/N比が多少悪くなりますが遅延検波(DPSK)を使い，位相の回転をキャンセルする方法があります．**図29**に遅延検波の処理例を示します．送られる情報は，一つ前のシンボルの位相と現在のシンボルの位相との差のみ意味をもたせます．位相の絶対値は，大きな意味はありません．受信の際には遅延検波によって位相の差分を取り出して復調します．その際に，マルチパスによる位相回転はキャンセルされます．筆者の場合は，処理が簡単なDPSKを変調方式として選ぶ場合がほとんどです．位相の微分が周波数なので，

〔図29〕遅延検波

この変調は位相変調というよりも周波数変調といえるかもしれません．

またマルチパスの影響は，周波数選択性のフェージングを発生させます．**図30**のように，マルチパスによってあるキャリア周波数の遅延した信号との位相関係がちょうど180°すなわち位相反転するとき，その周波数成分はキャンセルされて出力されなくなります．シングルキャリアの変調だと，その影響は全体に大きな影響を与えますが，OFDMの場合はそのキャリアにともなう一部の情報が欠落するだけです．OFDMの変調の際には，エラー訂正コードをともなった変調を行うのが一般的です．欠落した情報は，エラー訂正コードによって復元され，その影響を取り除くことが可能です．

はじめに述べましたが，とくに無線LANの環境では，この問題は深刻です．そのため，変調はOFDMもしくはBluetoothのように周波数ホッピングによる周波数拡散方式などが好んで用いられます．このような状況のため低速の無線モデムを除いて，シングルキャリアによる変調が用いられることはあまり聞いたことがありません．

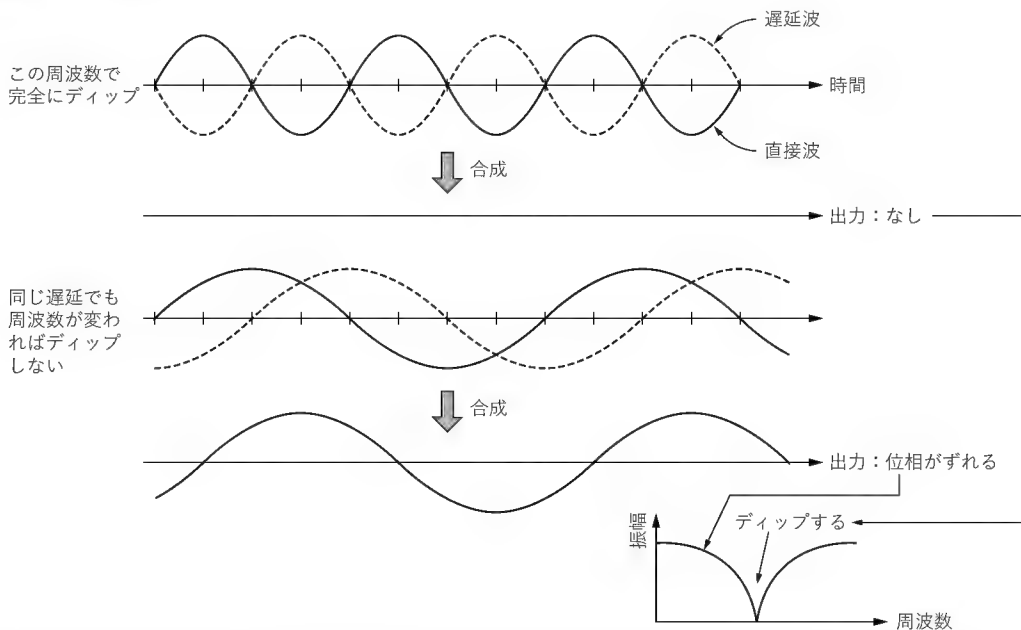
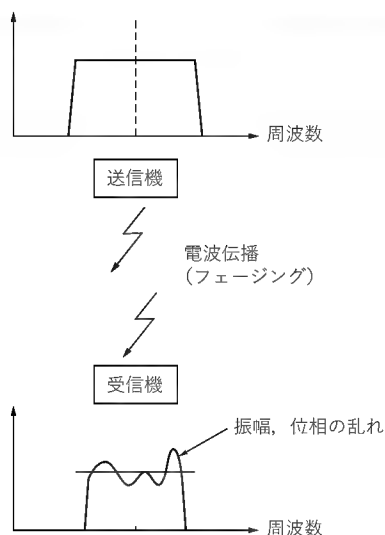
1.9 周波数特性劣化の影響

OFDMの長所として，各キャリアバンドは狭いため，周波数特性の劣化の影響を受けにくいと述べました．しかし，受信側の信号処理的には注意することがあります．送り側では，フラットなスペクトルでも，電波として空間を伝播してフェージングの影響を受けたり，無線機内の回路による特性の劣化で，**図31**のように受信側では，特性がフラットになることはまずあり得ません．

これは，受信側の処理で二つの悪影響を与えます．一つは，周波数特性の振幅特性や位相特性が劣化すると，せっかく送り

〔図30〕周波数選択性フェージング

〔図31〕周波数特性劣化

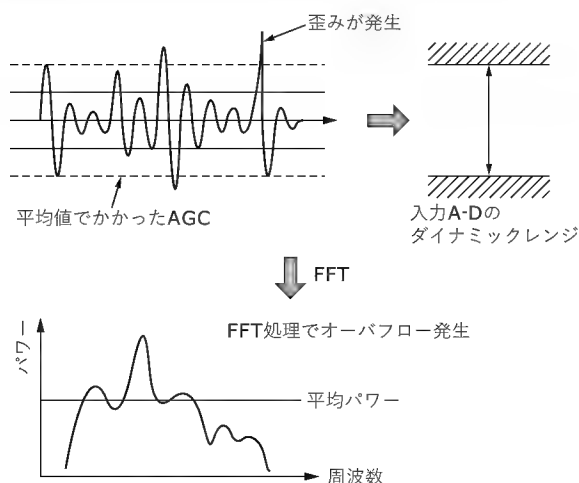
〔図32〕FFTとA-Dのダイナミックレンジ

側で*PAR*(Peak Average Ratio. 2.3参照)を抑えても，それ以上の比率の*PAR*の波形で受信される可能性があります．したがって，送り側の*PAR*に合わせて受信側のA-Dのダイナミックレンジを設計すると，問題を起こす可能性があります．信号の平均値でAGC(Automatic Gain Control)をかける場合も同じです．それらを考慮した，入力信号のダイナミックレンジの設計が必要です．

もう一つは，信号処理の問題です．OFDMを復調するときにFFTによって計算されます．FFTを固定小数点の処理系で設計する場合，送り側のフラットな周波数特性で各計算のダイナミックレンジを想定して最適化すると，問題を起こす可能性があります．**図32**のように，入力信号は全体でAGCがかかっています．そのため，周波数特性が劣化した信号をFFTにかけると，あるスペクトルの信号が特別大きなパワーをもち，予定したFFTのダイナミックレンジを越えることも予想されます．これは，直接的に重大なビットエラーを発生させます．そのことを予想した，FFTのダイナミックレンジの余裕ある設計が要求されます．

2 無線OFDMの設計例

2.1 劣悪な無線伝送路

ここで紹介するのは，短波帯の無線機に接続するOFDMデータモデムです．無線伝送において長距離伝播し，不安定な電離層状態に伝播による短波帯の無線伝送は，データ伝送にとってもっとも厳しく劣悪な伝送の一つです．無線LANでも状況は厳しいですが，頻繁に移動しなければ，特性は悪いながら安定しています．

しかし短波帯の伝送は，太陽活動にともない刻々とその影響を受けます．電離層から反射される電波はマルチパスあり偏波面の回転ありで，かなり深刻なフェージングを発生させます．レベル全体が時間とともに変動する成分や周波数選択性の変動もあります．短波帯のフェージングのモデルとしてよく知られているWattersonのモデルを**図33**に示します．このモデルでも実際をかなり簡略化したものと思いますが，それでもとても複雑な歪みを覚悟しなければならないことは明らかです．

このような状況では，LANと同じように複雑なマルチパスを考えた変調方式を使う必要があります．そこでOFDMによる変調を使うことにしました．UHF/VHFではこれまでシングルキャリアを使い，1,200bps程度の低速データ伝送がさかんに使われています．しかし，同じ手法は短波帯では使えません．

現在，世界的に短波放送や中波放送のディジタル化がDRMによって進められようとしています．2003年からは本格的な実験放送が始まりますが，もちろん変調方式はOFDMが採用されました．現在は**写真5**のように普通の受信機を一部改造して，その復調出力をパソコンのサウンドカードで取り込みソフトで音声に復元する，実験的なDRM受信機しかありません．写真ではパソコンは映っていませんが，台の下に隠してあり，LCDのディスプレイ部分だけ表示してある格好です．

しかし将来は大量に作られるLSIによって，現在のラジオを置き換える商品が現れてくるでしょう．ただし100円ショップでも売られているアナログラジオに対抗するのは，けっこうたいへんかもしれません．

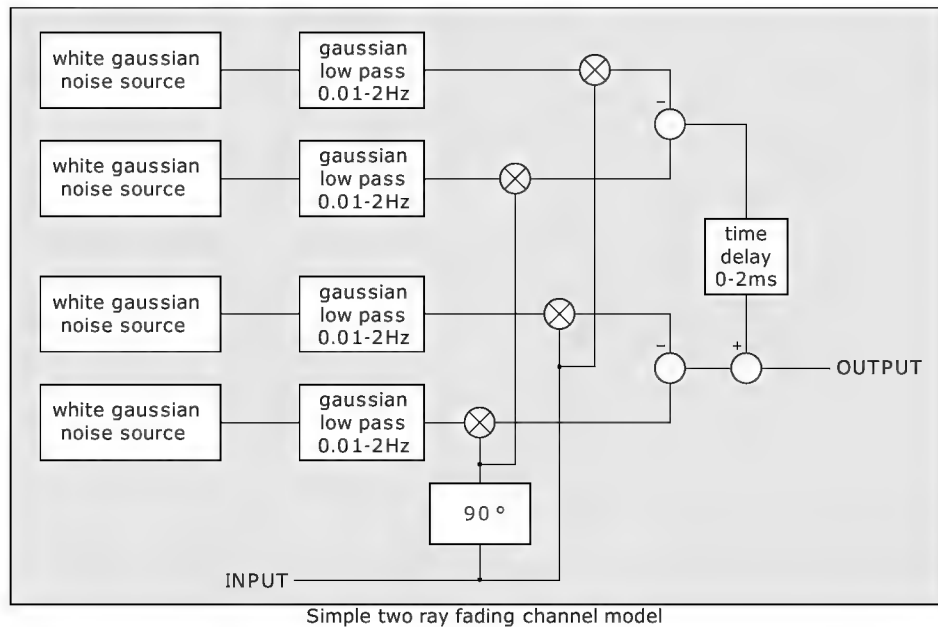
2.2 設計したOFDMの仕様

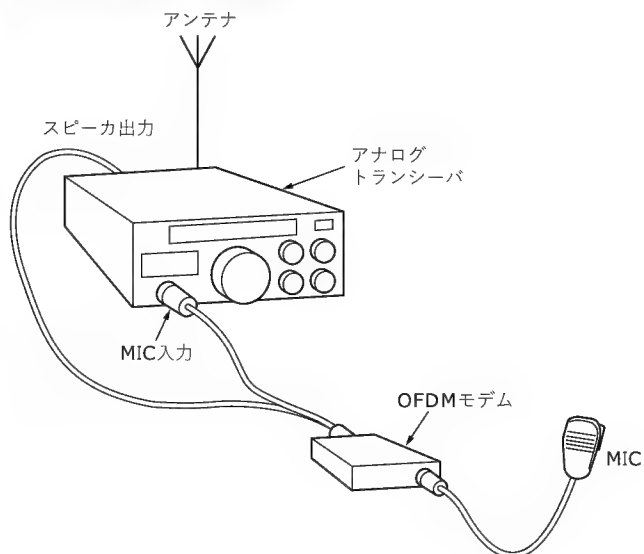
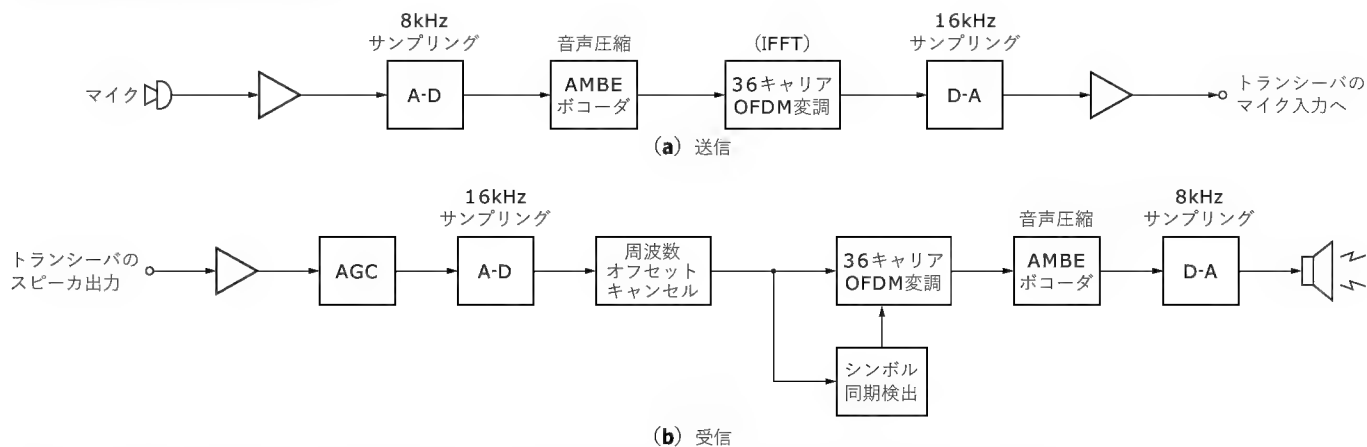
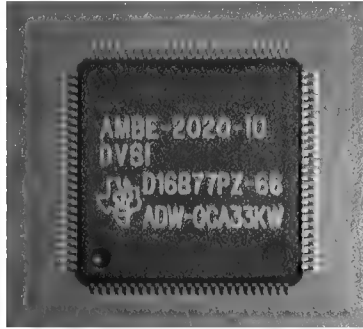
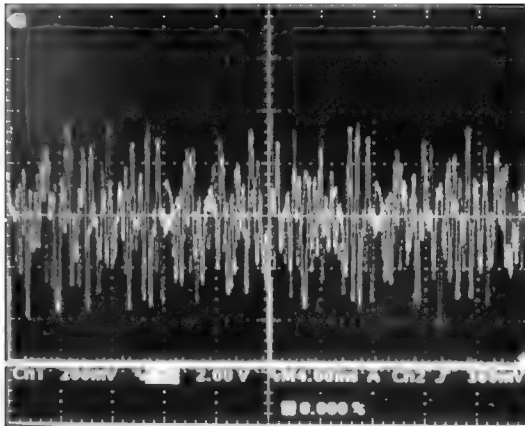
設計したOFDMのモデムは，**図34**のように従来のアナログ受信機のスピーカ出力，送信機のマイク入力を使ってディジタル音声伝送を実現するように設計しています．そのため，無線機とのインターフェースは音声領域のオーディオ信号となります．先ほどのDRMの実験受信機と同じです．

アナログ無線機の音声帯域はせいぜい300Hz～3kHz位程度しか確保できません．そこで，OFDMモデムの変調信号帯域を300Hzから2600Hzぐらいに選びました．

Information — 記録型DVDなどへのライセンスの開始
DVDの技術およびフォーマットの開発会社7社から構成されるDVD6Cライセンシングエージェンシーは，DVD-RAM/DVD-RW/DVD-Rなどの記録型DVDとDVD-Audioなどについての特許共同ライセンスを2003年1月より開始する．

〔図33〕WattersonHF フェージングモデル

Simple two ray fading channel model

〔写真5〕DRM受信機

〔図34〕無線モデムとの接続

〔図35〕OFDMモデムのブロック図

〔写真6〕AMBE2020

〔写真7〕OFDMの波形

全体のブロック**図35**を示しますが，一般的な8kHzでサンプリングされた音声信号をディジタルで伝送します．それに合わせて1シンボルを20msとしています．サンプリングされた音声はAMBEと呼ばれる優れた音声圧縮技術により2,400bpsという低速に圧縮されます．これにエラー訂正コードの1,200bps分を追加し3,600bpsを伝送しなければなりません．

それに合わせて36本のキャリアをそれぞれDQPSKで変調して，3,600bpsのスピードを得ています．音声ディジタル化のアルゴリズムAMBEはアメリカのDVSI社のAMBE2020チップを使っています．**写真6**のように，チップはTI社のDSPそのものです．

表1に全体の仕様をまとめてみました．短波帯ではもっとも厳しいマルチパスは2msくらいなので，4msのガードインターバルをとってあります．音声の場合はディレイが大きな問題となりますので，インタリーブは使っていません．そのため，基本的なOFDMのモデム部分を作ればいいことになります．

1）ヘッダ

ヘッダ部は，初めの同期を取るためとても重要です．ここでは**写真3**のような1000，1500，2000Hzの3トーンヘッダを使っています．それを1シンボルごとに位相反転(BPSK)し，同期を見つけやすくしています．

SSBの無線機も使うことを想定しているので，周波数オフセットの自動補償は±125Hzとかなり大きくとりました．それを可能にしたのがこのヘッダ部分です．短いヘッダの中で，周波数ずれを検出し自動補償します．

〔表1〕OFDMモデムの仕様

項　目		機　能
変調方式	OFDM変調	帯域300Hz～2.5kHz (HFトランシーバ) 36キャリア
	シンボルレート	20ms(50Baud)
	ガードインターバル	4ms
	トーン間隔	62.5Hz
	個別トーン変調方式 AFC	42キャリアの場合 8DPSK(6.3k)　4DPSK(4.2k) ±125Hz(とくにSSB通信向け)
エラー訂正	Voice：Golay+Hamming　Video/Data：Reed-Solomon	
インタリーブ	Voice：なし Video/Data：時間軸1秒，周波数軸 併用	
ヘッダ	1秒　3トーン＋BPSKトレーニングパターンによる同期用	
ARQ	CRCによる復号内エラーチェック エラー検出フレーム再送(データパケットモード)	
音声ディジタル	AMBEコーダ/デコーダ	
音声切替	ディジタル自動検出，自動切り替え	
映像圧縮	独自アダプティブJPEG準拠方式	
ビデオ	NTSC/またはPAL入出力	
電源	DC 10～16V　約170mA (DC6Vジャンパによる切り替え)	
COM	RS-232-Cレベル準拠　9600bps　非同期	
外形寸法	D：158×W：100×H：32mm	
コネクタ	RADIO	MIC出力(調整VR付き) SP入力(200mV～5Vp-p)　PTT
	VIDEO IN/OUT	NTSCまたはPAL 1Vp-p 75Ω
	MIC	MIC入力，SP出力(切り替え) PTT入力
その他	アナログ/ディジタル音声送信切り替えスイッチ 映像取り込み送信スイッチ	

2）DQPSK

前に説明したようにマルチパスによって，**図28**のような受信したコンスタレーションの位相回転は避けることはできません．そこで，差分4位相変調を使ってその影響を取っています．一つ前のシンボルとの位相差のみが，データとして意味をもちます．

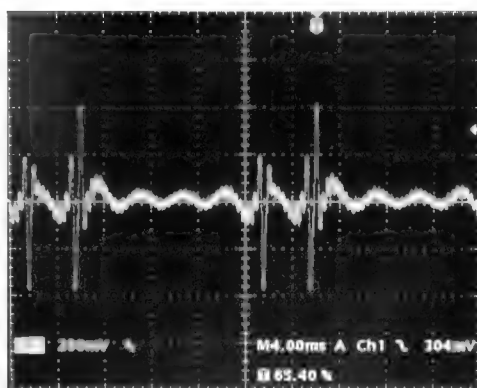
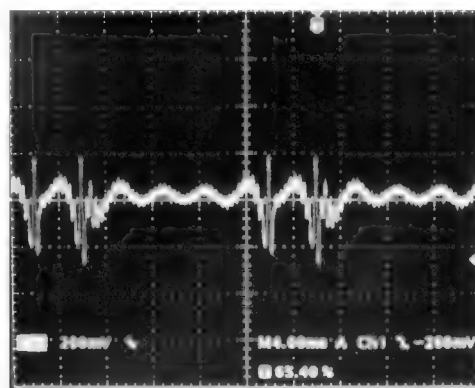
差分をとりますから，どちらかといえば周波数変調といえなくもありません．通常の同期検波に比べてS/N比は下がりますが，特性劣化時の特性自動補償やキャリア再生のPLLの実装の難しさを考えて，トータル的に考えて選びました．

3）スタートマーカ

OFDMの信号はランダムな白色ノイズのような特性をしています．その時本物の白色ノイズと見分けがつかないことが発生することが予想されるため，パケットデータ信号の始まりと終わりを示すマーカ信号を挿入しています．これを検出することにより，正規のパケットデータが受信されていると判断し，デコードすることが可能となっています．

2.3　送信機リニアパワーアンプの問題

OFDMのスペクトルはキャリアが隙間なく詰め込まれているので，**図23**のようにフラットな特性をしています．すなわち**写真7**のように白色ノイズと同じような波形をしています．いい

〔写真8〕36キャリアのOFDM

(a)

(b)

換えれば，信号のピークと平均値の比がとても大きくなります．この比を*PAR*と呼び，OFDMでは大きな問題となります．

1例として36キャリアのOFDMで，それぞれのキャリアの初期位相をすべて同じ位相にした場合で，初期位相の値をパラメータにいろいろと変えたときの逆フーリエ変換した波形を計算してみました．**写真8**の(**a**)と(**b**)に代表的な波形を示します．*PAR*がいろいろな初期位相で大きく変わることがよくわかります．

無線においてその飛びを決めるものは，平均値の送信電力です．たとえば5ワットの送信機でも，ピークの一時的電力が20ワットになったとすれば，波形をひずませないために20ワットの出力を出せるリニアアンプが必要になります．これは携帯無線機器においてはとても大きな問題です．**写真8**の(**a**)と(**b**)の平均値はほぼ同じです．この場合，ピークをひずみなく通すことが，とてもたいへんであることがよくわかると思います．

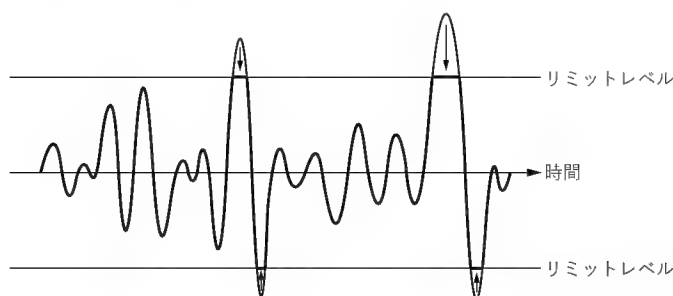
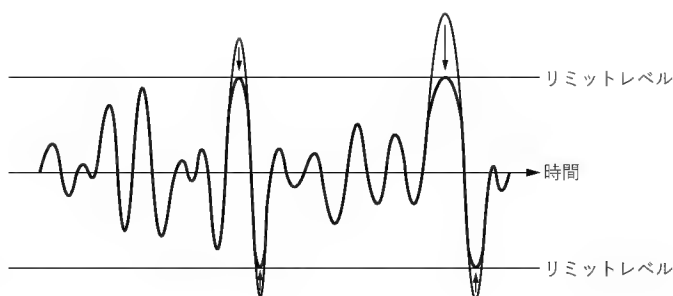
OFDMの変復調をアナログ処理だけで実現することはできません．そこで途中に必ずアナログ-ディジタル変換(ADC)やディジタル-アナログ変換(DAC)が含まれます．そこでダイナミックレンジを決めるのは平均値ではありません．ピークがカットされずに通過することが求められます．せっかく大きなビット数の高分解能のADCをつかっても，*PAR*の大きな信号をADCに通すと，その恩恵にあずかるのはピークのみです．多くの平均値は，MSB側がまったく意味をもたない，*S/N*比が悪いものになってしまいます．これはDACでも同じことがいえます．

このため，OFDMでは*PAR*を下げるために，何らかの処理をするのが一般的です．もっともスマートな方法は，変調するデータコーディングを工夫して，大きなピークを発生させないようにすることです．これは，ディジタル地上波や無線LANに採用されている方式です．*PAR*を6dB以内に抑えることが可能ですが，コーディングにより冗長なデータが増えるので，チャネルに余裕がある場合には有効です．詳しくは参考文献1)を見てください．

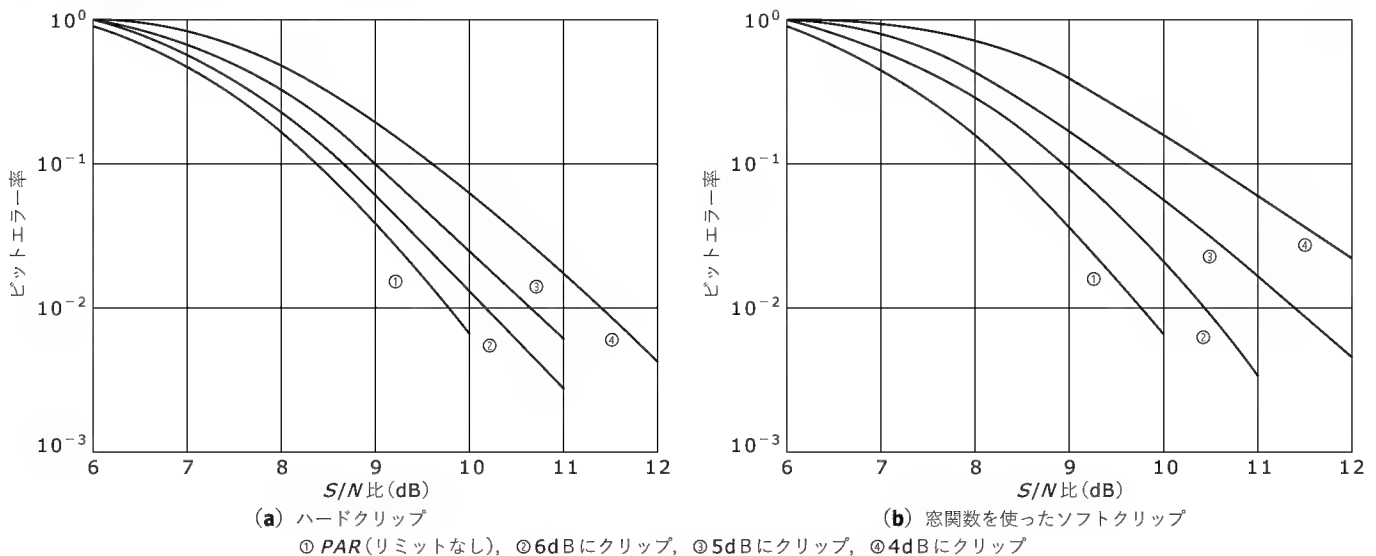
ここでは，もっとも簡単な**図36**(**a**)のようなピーククリッピングを使いました．クリッピングにもいくつかの種類があります．純粋にある一定レベルを超える信号を一定値に置き換える方法は，簡単ですがせっかく帯域内に抑えた信号をひずませるので，帯域外に不要輻射を発生させます．帯域を広げない方法として，**図36**(**b**)のように一般的なFFTを実施する前に行うような，窓関数を使って滑らかにクリップする方法が選べます．

いずれにしても信号をひずませますから，受信のときのビットエラーにどれくらい影響するかが気になります．**図37**に，ホワイトノイズを加えたときのビットエラー率を示します．予想に反して，窓関数を使うクリッピングよりも，単なるリミッタのほうがエラーレートは小さいことがわかります．

しかし，今度はクリップ後のスペクトルの広がりを考えてみ

〔図36〕クリッピング

(**a**) クリッパ　　(**b**) 窓関数を使ったリミッタ

〔図37〕クリッピングとビットエラー

(a) ハードクリップ　(b) 窓関数を使ったソフトクリップ
① PAR(リミットなし), ②6dBにクリップ, ③5dBにクリップ, ④4dBにクリップ

ます．単にクリップしただけでは，窓関数によるクリッピングに比べて，かなりスプリアスを発生することが予想されます．単なるクリッピングでは，信号のサイドローブは－40dBを越えてしまいます．このように，二つの方法の間には一長一短があり，目的に合わせて選ぶ必要があります．

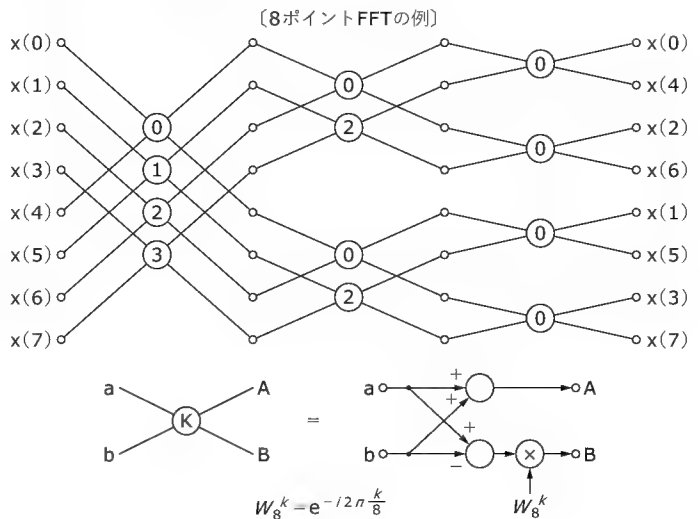
この応用では，すでにある無線機のマイク入力につなぐことが前提です．したがって，当然送信機のマイク入力は途中で何らかの帯域制限を受けます．なぜならば，マイクロフォンの出力は帯域制限された信号ではありません．マイクによっては10kHzぐらいの成分も含まれます．帯域制限がついていなければ，アナログ通信でも変調帯域が広がることになります．そこで，総合的に考えて純粋にクリッピングだけの処理にしました．

その他にも，ひずみを与えないでピークをキャンセルする方法があります．線形演算でキャンセルするので，帯域は広がりません．ただし，信号処理はとても複雑です．詳しくは参考文献7)を見てください．

2.4 逆FFTによる変調

OFDMの変調では離散的逆フーリエ変換を使って変調すると前に話しました．実際には高速フーリエ逆変換(IFFT)を使うことになります．**図38**のようなバタフライ演算を繰り返すFFTについては，これまで何度となくこの誌面にも登場していますし，ディジタル信号処理の代表格みたいなものです．

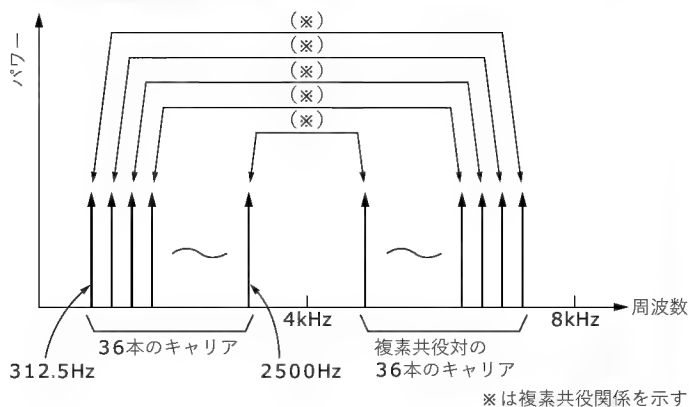
ご存知のように，FFTでは変換する区間のサンプル数は2のべき乗にかぎられています．ここの設計では，8kHzサンプリング128ポイントのFFTを使うことにしました．したがって，いちばん低い周波数の基本波は62.5Hzになります．

それぞれのキャリアはQPSKなどの直交変調をするので，当然入力は複素数となります．すなわち128個の複素数から，実数のOFDM変調波形を作り出すことになります．この設計では，OFDM後の2次変調は直交変調ではありません．すなわちIFFTした出力が実数とならなければなりません．そのため**図39**のように，後半のキャリアは前半のキャリアと対の関係で共役複素数となります．つまり36本のキャリアを使うことは，それと対になった共役な36本のキャリアも必要になります．そこで128本のキャリアのうち，**図39**のように72本のキャリアの入力を設定することになります．しかし実際は，半分は共役複素数入力ですから，36本を計算すればいいことになります．

〔図38〕FFTのバタフライ演算

1) SH-2によるFFTの計算

はじめに，このモデムの中で処理されているディジタル信号処理(DSP)の流れを**図40**に示します．これらの処理は，必ずしもソフトだけで処理されているわけではありません．半分くらいは，FPGAのハードウェアDSPで実現しています．プロセッサの信号処理にもう少しパワーがあれば，すべてソフトDSPで

〔図39〕複素共役対のキャリア設定

〔写真9〕内部基板

できたかもしれません．ただ，デバッグのやりやすさからいえば，FPGAの処理のほうが使いやすいと感じました．**写真9**にこのモデムの内部基板を示します．写真で示すように，AlteraのFPGAによるDSPとSH-2によるソフトDSPとの協調設計となっています．

IFFTの実装は，DSPではなくSH-2のソフトで行いました．SH-2は内部のタイマで62.5 μs(16kHz)の割り込みがかかっています．その割り込みに同期して，OFDMの入出力は16kHzでサンプリングされています．

一方音声のCODECは8kHzなので，**図35**のように入力は16kHzでサンプリングされた信号は，半分の8kHzにサブサンプリングされます．また出力は逆に，IFFTが出力する8kHzの信号を2倍にオーバサンプリングしています．それぞれ**図40**のようなサブサンプリングFIRフィルタとオーバサンプリングFIRフィルタをSH-2で実現しています．このように倍のサンプリング周波数を使うのは，ADCの前やDACの出力につくアナログフィルタの構成を簡単にしたかったからです．

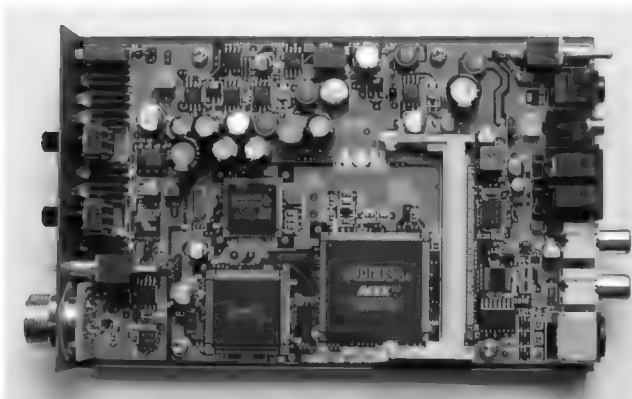
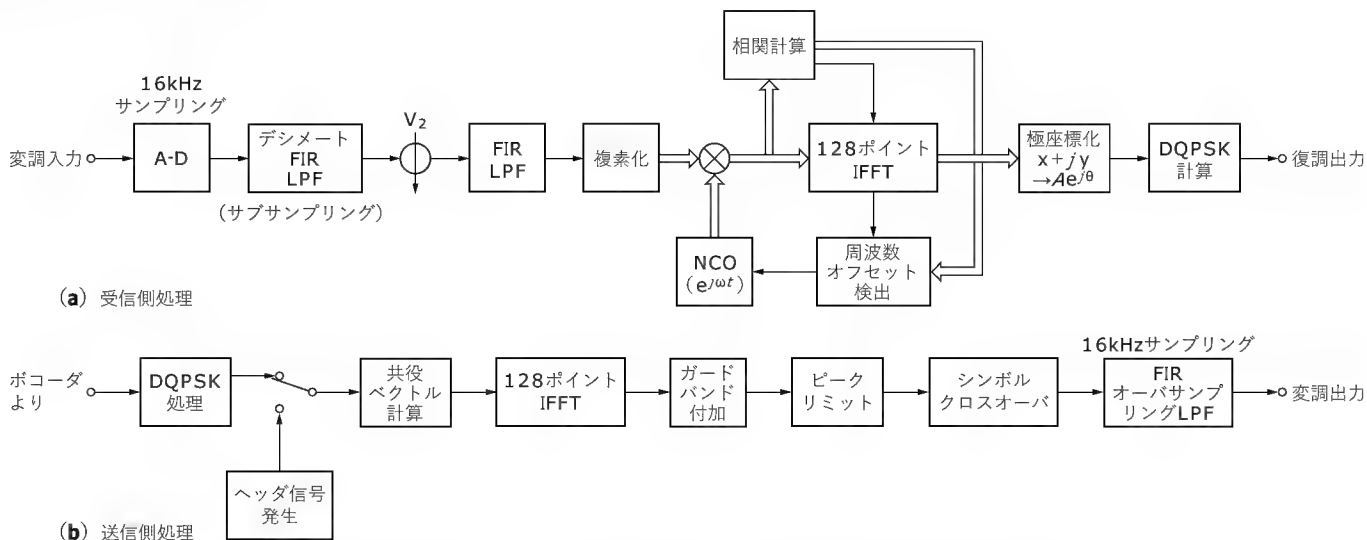
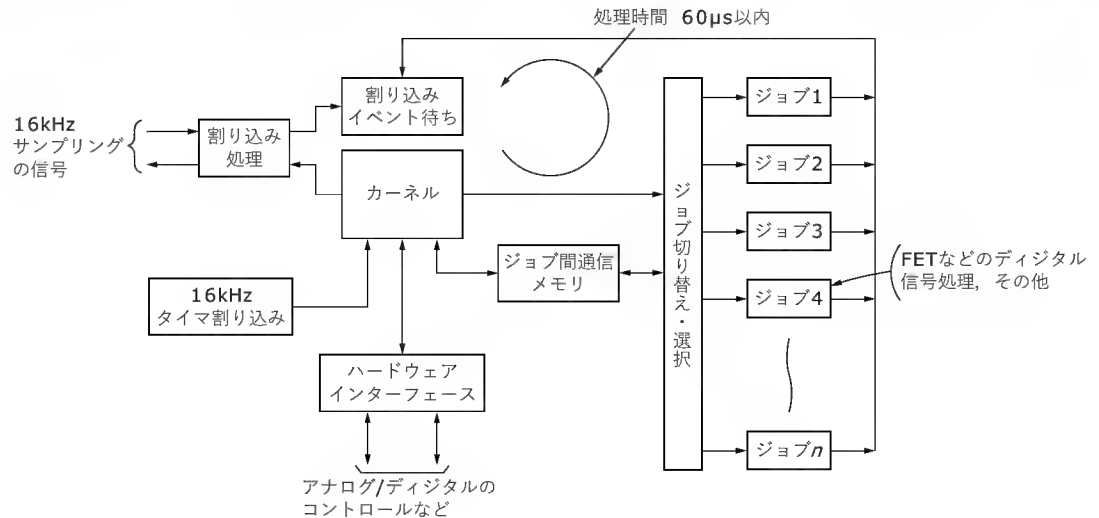
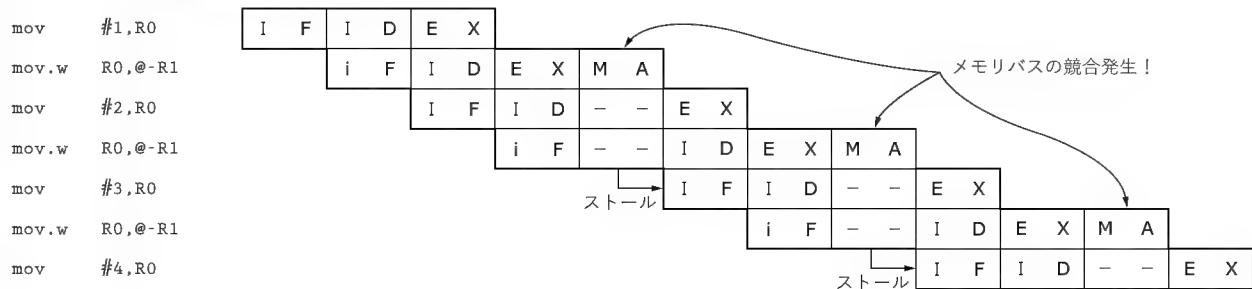
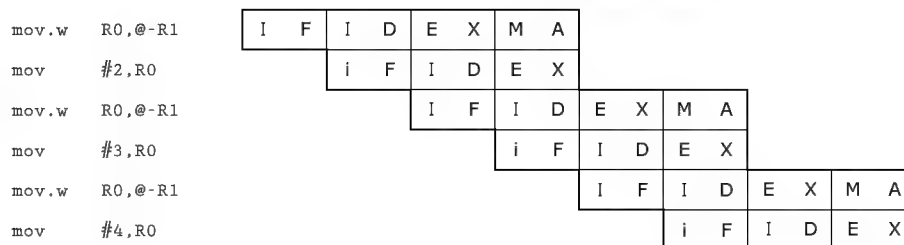
IFFTを計算するときも，**図41**のようにこの62.5 μs単位に仕事を分割しています．それぞれのモジュールは必ず62.5 μs以内で処理が終わるように計算されています．リアルタイムOSのようにジョブが並列に走る場合は，OSのほうで各ジョブに対してレジスタの連続性を保障しなければなりません．すなわちジョブの切り替えに対して，かなりのオーバヘッドが必要なため，このように62.5 μsという短いサイクル時間単位で，多くのリアルタイムディジタル信号処理を行うには適さないと判断しました．

実際には，**図41**のような簡単なカーネルによって処理をコントロールしています．各ジョブがカーネルに処理の終了を知らせるときは，必ずそのジョブが起動してから62.5 μs以内ということを暗黙の了解にしました．そこでレジスタの保護はまったく行っていません．つまり62.5 μs以上かかる処理は複数の短いジョブに分割します．しかもすべての割り込み源は，カーネルの部分でしか許可されないので，各ジョブの作業中に割り込みがかかることはないように設計しました．また**図41**のように，ジョブ間の通信はすべてメモリを介して行われます．

図38にIFFTの有名なバタフライ演算を示しましたが，概略1段のバタフライ演算を二つのジョブすなわち125 μsで計算し

〔図40〕DSPのブロック図

〔図41〕プログラム構造

ています．筆者の本の中にも書きましたが，SH-2を使い高速にディジタル信号処理処理を行うためには，SH-2のパイプライン処理の流れを理解しておく必要があります．SH-2はハーバードアーキテクチャではないため，**図42**のようにバスの競合が発生し，片方のデータ処理は待たされることになります．具体的にいえば，パイプライン処理の構造のため命令のフェッチとデータメモリの入出力で衝突が発生します．命令実行のためのメモリフェッチは必ず必要ですが，16ビットのRISC命令を32ビットバスで一度に二命令をフェッチします．そのため，バス上で命令をフェッチしないサイクルが交互に発生します．このときにできるだけデータの入出力の実行ユニットがパイプライン上に来るように命令を並べればいいことになります．

具体的なFFT処理の例は説明が長くなるので，ここでは省略しますが，筆者の著書[6]に具体的に記したので，それを参考にしてください．FFTの演算はもちろん固定小数点の演算ですが，その桁取りがもっとも難しいノウハウです．FFTの場合はフーリエ変換を高速化するために，各128データが同じ数の掛け算を通過しません．これが固定小数点でのFFTの計算を困難にしています．加減算の場合は，比較的桁どりはたやすいのですが，掛け算を使うととたんに複雑になります．

〔図42〕メモリアクセスの偶数配置

(**a**) 奇数配置の場合

(**b**) 偶数配置の場合

I F：実際にメモリフェッチをともなうインストラクションフェッチ
i F：メモリフェッチをともなわないインストラクションフェッチ
I D：インストラクションデコード
E X：命令実行ユニット
MA：データメモリアクセスユニット

2）変調データの生成

OFDMの変調データを作ることはいたって簡単です．OFDMの入力は36組の複素数を設定することです．**図43**のように虚数部をQ軸，実数部をI軸としたQPSKを考えればいいことになります．それぞれのキャリアを直交変調することはすなわち，それぞれのキャリアに対する複素数を設定することになります．QPSKだと$(I,\ Q)$とすると，$(a,\ 0)$，$(0,\ a)$，$(-a,\ 0)$．$(0,\ -a)$のいずれかの値をとります．または，それぞれを任意の角度で複素平面上回転させた値でもかまいません．ちなみに$\pi/4$だけ回転させた場合は，**図43**のように$(a/\sqrt{2},\ a/\sqrt{2})$，$(-a/\sqrt{2},\ a/\sqrt{2})$，$(-a/\sqrt{2},\ -a/\sqrt{2})$．$(a/\sqrt{2},\ -a/\sqrt{2})$となります．

aは任意の振幅です．しかし，aの振幅の値を決めることはけっこう難しい作業です．IFFTを使って実時間の信号波形に変換したときに，波形のピークが固定小数点演算のダイナミックレンジのオーバフローを起こさないような値を選ばなければなりません．どのキャリアに対しても同じ値を使うのが普通ですが，キャリアによって振幅を変えることも可能です．たとえば，とくに重要なデータを送るキャリアのパワーを大きくとることも可能です．また，伝送路の周波数特性の劣化をあらかじめ補償するために自由にプリエンファシスをかけることも可能です．あるいは，**図44**のようにすでに存在する信号を避けるために，前から存在する信号との干渉の恐れのある周波数のところだけ空白にすることもできます．

ボコーダAMBE出力は，圧縮された音声データとエラー訂正コードで，1シンボルあたり72ビットです．それをまず2ビットずつ区切ります．36個の2ビットペアができます．DQPSKなので，**図45**のように一つ前のそれぞれのキャリアの位相，すなわち複素数値からのそれぞれのキャリアに対応するデータ2ビットで位相回転し作ります．DQPSKの場合，いちばんはじめのシンボルは値が不定になります．前の位相がわからないからです．1番目は参照となる基準位相を決めるだけの働きしかしないので，データと分ける意味で，リファレンスベクトルと呼ばれています．

残り36個のキャリアのデータは，計算した36個のデータの共役複素数を計算するだけです．簡単にいえば，虚数部の極性を逆にするだけです．128個の入力データの内残りの使わないものはゼロをいれて，出力しないようにします．こうして作った128個のデータを先ほどのIFFTにかければ，OFDMの信号が得られます．

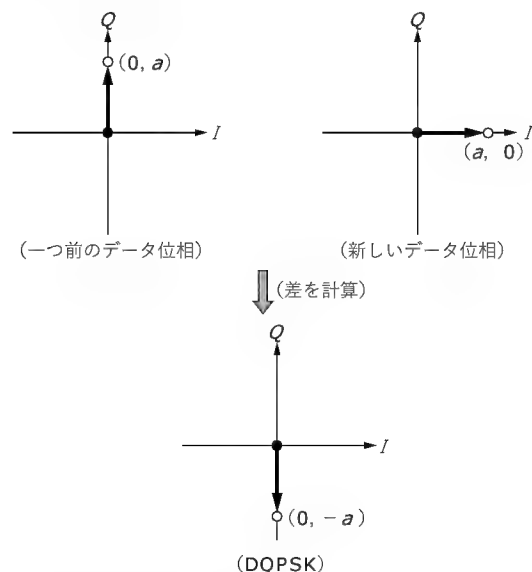
2.5 ガードインターバルの挿入

128点のIFFTをすると$125\mu s \times 128 = 16ms$の信号が得られます．これに4msのガードインターバルを付加します．**図46**のように全体のシンボル時間は，設計値の$16 + 4 = 20ms$となります．ガードインターバルの信号を付加する処理は，とても簡単です．**図18**のようにIFFTで計算した信号の後ろの4msをカット&ペーストすればよいだけです．

〔図43〕QPSKのコンスタレーション

〔図44〕スペクトルの自由度

〔図46〕OFDM1シンボル

〔図45〕DQPSK

IFFTは同じ信号が繰り返すことを前提に，無限積分のフーリエ変換を有限区間に限定して計算しています．このカードインターバルを追加する処理は，この理屈にかなっており，つなぎ目の不連続性はありません．すなわち，これによって有害なスプリアスは発生しません．

むしろ問題なのは，シンボル間の結合です．シンボル間の信号の相関はまったくないので，つなぎ目が不連続になってしまいます．これにより有害なスプリアスを発生しています．この対策としては，つないだ後にフィルタに通して不要な信号を落とすことがまず考えられます．しかし，広帯域のOFDMの信号だけ通過させるようなフィルタは，けっこうたいへんな処理になることが予想されます．

もっと簡単な処理で，不要な信号を発生させない方法があります．隣り合う相関のないシンボル同上を突然切り替えるのではなく，**図47**のように徐々にフェードIN，フェードOUTしてクロスオーバするやり方です．具体的には，それぞれの信号を足したときの振幅が一定でなければなりません．そのため，ハニング窓のように，中心を境に上下対象の波形を掛け合わせたあとに足し算をして処理します．

この効果は絶大です．1シンボル160サンプルに対し，何も処理しないで接続すると信号帯域の4倍のサイドバンドのレベルでも－40dB以下になることはありません．一方わずか4サンプルの窓関数処理でも，2倍のサイドバンドでもレベルが－50dBまで低下します．何もしない特性に比べて劇的な効果があることがわかります．ここの設計では実際に4サンプルの窓関数処理を行っています．そのため，受信側でFFT処理して復調する際には，この4サンプル分を避ける必要があります．ガードインターバル32サンプルのうち4サンプルをサンプルの後半に設けることによって，これに対応しています．**図47**のようにプリおよびポストガードインターバルを設けています．

2.6 周波数ずれの補正

ここで使っているOFDMでは，キャリア間隔は62.5Hzです．とても狭い間隔でぎっしりとキャリアが詰まっています．前にも話しましたが，受信した信号約30Hz以上の周波数オフセットがあると，キャリアの直交性はまったく崩れ，シンボル間の干渉のため，ほとんど復調できなくなります．

ところがSSBなどの信号で通信する場合は，30Hz位のずれは普通に発生します．そのような伝送路が使われることを考えて，前に説明した特別の3トーンヘッダ信号を使って，周波数ずれを検出し自動補正をかけます．また時間軸のシンボル同期も，前に説明したようにヘッダの特徴を生かしてかけます．

3トーンヘッダによって同期が高速に確立したあとは，**図26**のようなガードインターバルの相関性を使った同期保持が働きます．具体的なしくみは前に説明しました．時間軸の同期は使い慣れた，またよく耳にするPLLを使えばいいことになります．周波数の補正は，**図48**のような処理により周波数変換し行われます．

この処理は複素周波数ヘテロダインといわれます．まずは計算式で，どのような処理が行われているかを示します．まず入力した信号をFIRヒルベルトフィルタに通し，その信号遅延分のディレイラインを通過した2種類の信号に変換します．この信号はそれ以降，複素数として処理されるので，この処理のことを信号の複素化と呼んでいます[6]．

複素数正弦波　$Y = Ae^{j\omega t} = A\cos\omega t + jA\sin\omega t$

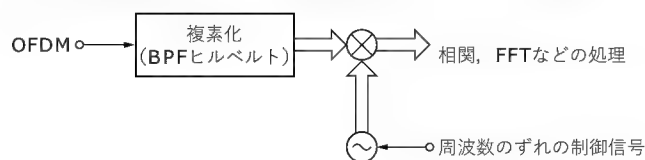
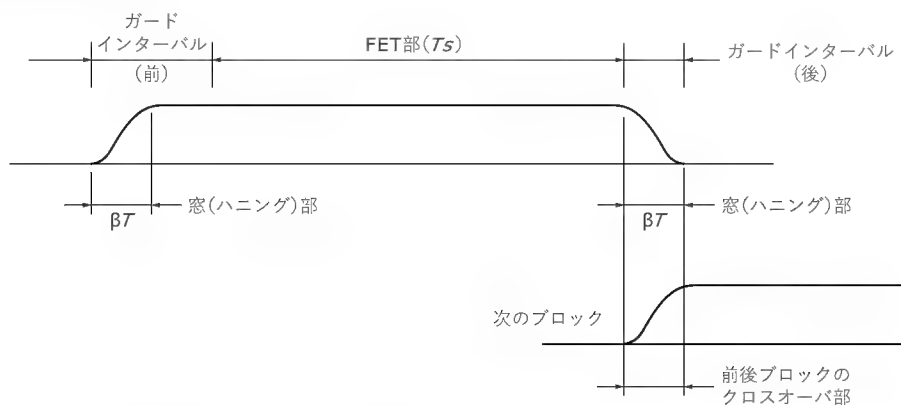
複素ヘテロダイン

信号　$S = re^{j\varphi t} = Sr + jSi$ （Si：ヒルベルト変換出力）

キャリア　$-C = e^{j\omega t} = \cos\omega t + j\sin\omega t$

ヘテロダイン出力　$Y = S \cdot C = re^{j(\omega + \varphi)t}$

複素可変周波数発振器が必要になります．アナログのVCOの

〔図48〕複素ヘテロダインによる周波数オフセットキャンセル

〔図47〕窓関数によるスプリアス対策

ディジタル信号処理版で，入力される数値によって出力の周波数が変わるものです．複素数ですから，直交する二つの正弦波を発生させなければなりません．sinとcosの関係です．それを式に沿って極座標形式で書くと，もっとはっきりします．複素平面を反時計方向に回転するのが正の周波数で，周波数変換後は元の周波数よりも高い周波数に変換されます．時計方向に回転するのが負の周波数で，変換後は元の周波数より低い周波数に変換されます．

複素化された入力信号と，周波数ずれに対応して発生させた複素正弦波との複素掛け算をすれば複素ヘテロダインが計算できます．式で書けばとても簡単ですが，実際に計算する場合は信号の複素化がもっともたいへんな処理となります．とくに複素化には欠かせないヒルベルトフィルタを実装するのがけっこう重い処理です．なおヒルベルトフィルタの設計に関しては，設計ツールをもっていない場合は，筆者が書いた著書中[6]の設計の要点を参考にして，付属CD-ROMにある設計ソフトで設計して試してみてください．

それ以降のOFDMに必要なのは実数部だけですが，**図26**に示すようにガードインターバルの相関計算の際に共役複素数を計算する必要があり，複素ヘテロダインの複素計算出力がそのままその計算に使われます．

複素ヘテロダインの処理は，SH-2ではなくAltera ACEXによるハードウェアで実現しています．処理は**リスト1**のようなVHDLですべて記述しています．もちろんソフト処理でも実装可能ですが，DSPと違ってSH-2の処理能力はあまり余裕がないため，できるだけ負荷をかけないようにハードウェアで実装しました．

この設計した商品のハードウェアは，**図49**のようにソフト・ハードの境目をなくし，都合に合わせて各機能・処理系はどちらでも実装できるようにしています．AlteraのFPGAは10万ゲート相当を使っていますが，その配置配線データはすべてSH-2のほうからダウンロードするようなしくみにしてあります．SH-2

〔リスト1〕VHDLサンプル

```
-- pai/2 rad SIN table ---------------------------------------------
--       (C)2002 copyright Y Nishimura
------------------------------------------------------------------

LIBRARY ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
LIBRARY lpm;
use lpm.lpm_components.all;

entity Sintab is
    port
        (
            Mclk      : in    std_logic;  -- Master clock
            Reset     : in    std_logic;  -- Master Reset
            wrst      : in    std_logic;  -- Reset table address
            wden      : in    std_logic;  -- Write enable
            radd      : in    std_logic_vector(9 downto 0);
            RamIn     : in    std_logic_vector(15 downto 0);
            Sindata   : out   std_logic_vector(15 downto 0)
        );
end Sintab;

ARCHITECTURE one OF Sintab IS
    signal  Mdata           : std_logic_vector(15 downto 0);
    signal  wadd,Rradd      : std_logic_vector(7 downto 0);
    signal  wsta            : std_logic_vector(1 downto 0);
    signal  Te,wen          : std_logic;

BEGIN
-- Memory Access Sequence ------------------------------------------
--
------------------------------------------------------------------
    process(Mclk,Reset)
    begin
        if Reset = '0'  then wsta <= "00";
        elsif rising_edge(Mclk) then
            case wsta is
                when "00" => if wden = '1'  then wsta <= "01";
                                            else wsta <= "00";
                             end if;
                when "01" => wsta <= "11";
                when "10" => wsta <= "00";
                when "11" => if wden = '0'  then wsta <= "10";
                                            else wsta <= "11";
                             end if;
                when others => wsta <= "00";
            end case;
        end if;
    end process;
    wen <= (not wsta(1)) and wsta(0);
------------------------------------------------------------------
--      Memory write pointer
------------------------------------------------------------------
    process(Mclk,Reset)
    begin
        if Reset = '0'  then wadd <= "00000000";
        elsif rising_edge(Mclk) then
            if wrst = '1'       then wadd <= "00000000";
            else
                if wsta = "10"  then wadd <= wadd + 1;
                                else wadd <= wadd;
                end if;
            end if;
        end if;
    end process;

------------------------------------------------------------------
--      Read address generator
------------------------------------------------------------------
    process(Mclk,Reset)
    begin
        if Reset = '0'  then Rradd <= "00000000";
        elsif rising_edge(Mclk) then
            if Radd(8) = '0' then Rradd <= Radd(7 downto 0);
                             else Rradd <= not Radd(7 downto 0);
            end if;
        end if;
    end process;

-- Dual port ram -- 256 words ---------------------------------
    Te <= '1';
    U1: lpm_ram_dp
        GENERIC map (LPM_WIDTH => 16, LPM_WIDTHAD => 8)
        port map (rdaddress => Rradd, wraddress => wadd,
                  rdclock => Mclk, wrclock => Mclk, wren => Wen,
                  wrclken => wen, rdclken => Te, rden => Te,
                  data => Ramin, q => Mdata);

------------------------------------------------------------------
--      Output data control
------------------------------------------------------------------
    process(Mclk,Reset)
    begin
        if Reset = '0'  then Sindata <= "0000000000000000";
        elsif rising_edge(Mclk) then
            if Radd(9) = '0'    then Sindata <= Mdata;
                                else Sindata <= (not Mdata) + 1;
            end if;
        end if;
    end process;

END one;
```

〔図49〕ソフトウェアとハードウェアの融合

ソフト/ハードのデータをフラッシュメモリにダウンロード
PC
SH-2 フラッシュメモリ
ハード
Altera CPLD 10万ゲート
シリアル
SH-2のフラッシュメモリにあるネットデータでコンフィグレーション

- 基板上のハードウェアは，その定義にデータをダウンロードしなければ，何も動作しない
- ソフト設計とハード設計を同時にインタラクティブにできる

のROMはフラッシュメモリになっており，ソフト/ハードを含めた設計コードは外部のシリアルインターフェースを通して，自由に書き換えできるように柔軟性をもたせてあります．

このような構成にしてから，これは筆者の好みの問題かもしれませんが，ハードで実現できるものは，ハードウェアで実装する傾向にあります．しかし，設計自体は回路図ではなくVHDLの記述で行うので，見た目はすべてソフト設計のようにも見えます．

2.7 NCO

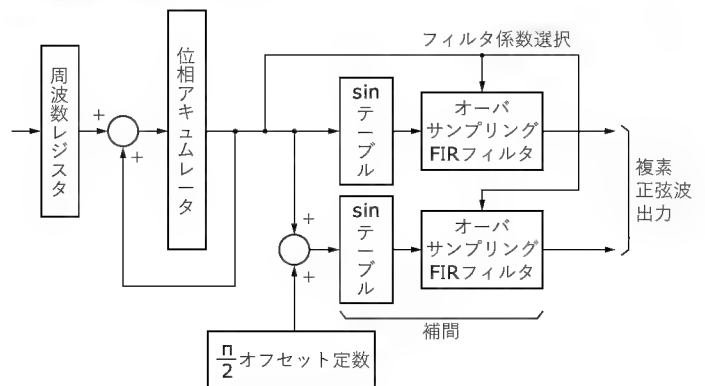
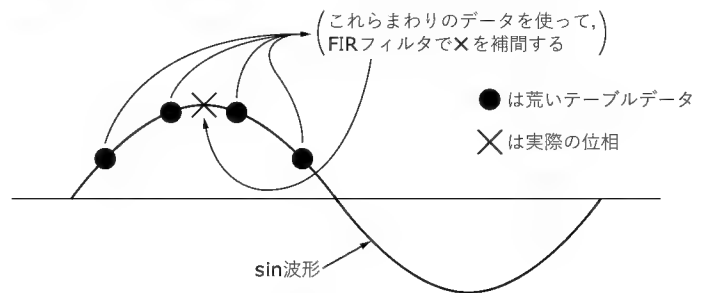
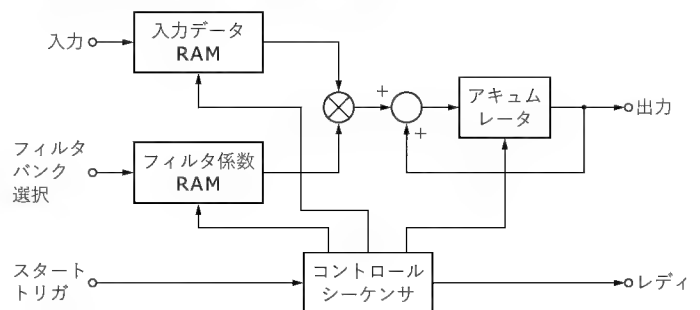
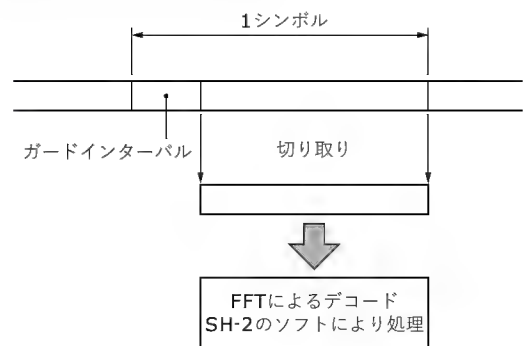
ここで使う複素発振器の実現には，いろいろな方法があります．ここでは周波数ゼロ(DC)を含むプラスマイナスの広範囲な周波数を生成しなければなりません．そこで，安定性があり構成が簡単な**図50**のような位相アキュームレータとテーブル参照による方法で実現しています．ただし，出力の波形の精度はテーブルの細かさで決まります．

大きなテーブルを使えばいいのですが，その実装のために大きなメモリが必要で，現実的ではありません．そこで，**図51**のように荒いテーブルとオーバサンプリングによる補間で精度を高めています．オーバサンプリングフィルタはFIRの構成で，**図52**のようにとても少ない計算量で実現できます．

ハードウェアによるFIRフィルタは，FPGAのクロックに比べてそんなに高速性は要求されません．そこで，**図52**のように掛け算器を1個と積算器を1個もつハードウェアで，係数1個ずつ計算するステートダイアグラムを組み実現しています．sinテーブルやFIRフィルタの係数は，AltaraのFPGAがもつ埋め込みRAMを使って格納しています．もちろん初期値はSH-2のほうからダウンロードします．

2.8 FFTによる復調

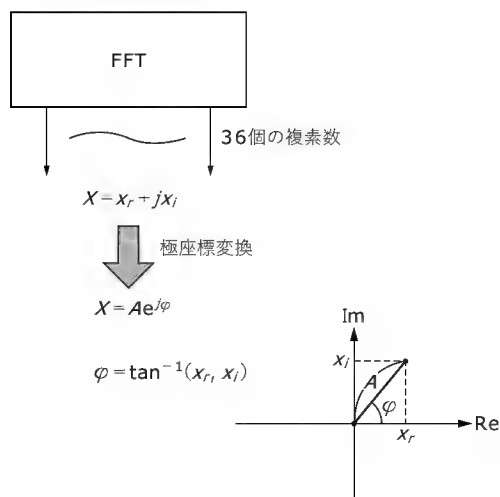
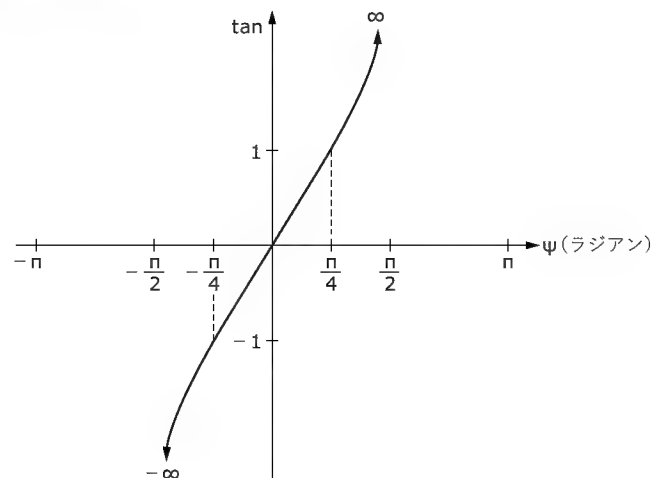
周波数オフセットも補正し，シンボル同期が確立したあとは，FFT処理をすることでデータの復調を行います．FFT処理は，変調のときに使ったIFFT処理とほとんど同じ処理で，共通に使える部分が多くあります．実際にサブルーチンの形で使っています．**図53**のように汚染されたガードインターバルを除いた純粋な128個のデータを取り出し，FFTにかけます．

〔図50〕複素NCO

〔図51〕テーブルの補間

〔図52〕シーケンシャルFIRフィルタ

〔図53〕FFTによるデコード

通常のFFT処理の前に行う偽信号を取るための窓関数をかけることは，OFDMのFFTにおいては必要ありません．もともとOFDMは1シンボルという区間に区切られた，全体では連続性のない信号を送ることで伝送しています．区切りがシンボル

同期ではっきりしているため，そのまま切り出してFFTを掛ければ正確に復調できます．

この処理も，変調側と同じくSH-2のソフト処理で実装しています．62.5μsのタイマ割り込みを基準に62.5μsで処理が完結する細かなジョブに区切り，処理します．128点のFFTを計算しますが，必要なのは36個のデータだけです．計算結果は当然複素数になり，それぞれのキャリアの直交復調した結果が得られます．キャリア1個1個を直交復調しようと考えると，とてもたいへんな処理に思えますが，FFTを使うと比較的簡単に復調できるのに驚きます．

2.9 複素極座標変換とPSKの復調

このOFDMモデムではDQPSKを使っているため，位相角のみがデータ処理に使われ，信号の振幅成分には大きな意味がありません．逆にフェージングやAGCなどによって，振幅成分には多くのひずみ成分が含まれます．そこで，**図54**のようにまず受信した36組の複素数を直交座標から極座標に変換します．極座標変換することにより，振幅成分と位相角成分とにはっきり分けられます．

そのため極座標に変換する際に，振幅成分はその後の処理には使いません．そこで位相角のみの計算を行います．位相角は**図54**のように，arctanを計算すればいいことになります．tan(正接)の関数は**図55**のように$\pi/4$ラジアンを越えたところから急速に発散してやがて無限大になります．ここでは，テーブル参照によるarctanを計算するので，精度を上げるためできるだけダイナミックレンジは抑えたいところです．

そこでまずは複素平面を$\pi/4$ラジアンごとに8個の領域に区切り，どこの区分に入るかを調べます．次に**図56**で奇数領域の場合は普通に，Y＝虚数部(X_i)/実数部(X_r)の絶対値を計算します．偶数領域にある場合はY＝実数部(X_r)/虚数部(X_i)の絶対値を計算します．すなわちYは1以下になり，tanの関数の発散の影響を受けないで，ダイナミックレンジを小さく抑えることができます．

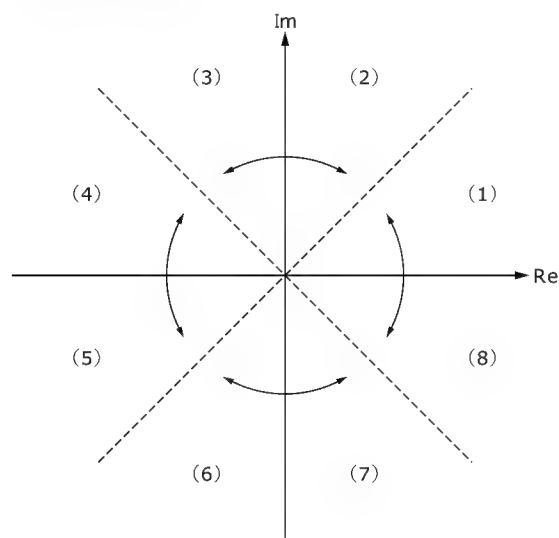
割り算のハードウェアはDSPでもないのが一般的です．しかし割り算のための命令は用意されています．ここではSH-2を使いますが，**リスト2**に割り算の処理を示します．一般的なソフトウェアの処理では，できるだけループを用いて簡明にかつ短く記述しようとしますが，ディジタル信号処理では見た目の美しさより，処理時間がもっとも重要です．そのためループを使わずに同じ命令をずらっと並べる処理形式をとっています．

計算した結果を使って，arctanのテーブルを引き角度に変換します．変換された角度は0～π/4ラジアンになるはずです．まず初めに，複素平面状のどの領域に入るかを調べました．それによって最終的なarctanの結果0～2πラジアンを得ることができます．

一つ前のシンボルのときも同じようにして計算した位相角のデータが保存してあります．その値と，計算した結果との差を

〔図54〕極座標変換

〔図55〕tan関数

〔図56〕$\tan^{-1}$の区分け

〔リスト2〕割り算処理

```
;----- Divide routine ------------------------------------
;       R0/R2
;---------------------------------------------------------
DIVIDE: mov     R2,R4           ; Save data
        cmp/pz  R4
        bt      DIVIDE2         ; If plus data
                neg     R4,R4
DIVIDE2:
        xor     R0,R2           ; Set Sign data
        cmp/pz  R0              ; Test sign
        bt      DIVIDE1         ; If plus data
                neg     R0,R0
DIVIDE1:
        shll16  R0
        shll16  R4
        div0u
    .AREPEAT 15
        div1    R4,R0
    .AENDR
        rotcl   R0
        extu.w  R0,R0
        cmp/pz  R2              ; Test sign
        bt      DIVIDE3
        neg     R0,R0
DIVIDE3:
        rts
        nop
```

〔図57〕信号処理のリアルタイム性

〔図58〕インタリーバ

計算すれば，最終的なDQPSKの復調データとなります．位相角の計算はけっこう重い処理ですが，1シンボルにつき1回計算すればいいので，時間的な余裕はあります．

2.10 もっとも重要なシンボル同期維持

これで，ひととおりのOFDMモデムの処理を説明しました．これまでの説明で，OFDMの処理がけっこう簡単にできることがわかったと思います．もっとも複雑な処理は，FFTの計算です．これを自前で作る場合は，ちょっと気合を入れて作る必要があります．しかし，ある程度それぞれのDSPまたはCPUに対して，メーカーごとに標準ライブラリとしてすでに用意されていることも多いと思います．もしそれを使うことができれば，比較的短時間に作ることができるでしょう．

処理系を作る場合は，普通のソフトの設計と違い，リアルタイム性を重視したディジタル信号処理であることを常に頭に置いておく必要があります．リアルタイムOSを使う場合も多いと思いますが，**図57**のようにμs単位のけっこう短い間隔でサンプリング周期が発生するため，この処理を最優先にして時間的なゆらぎが発生しないようにしなければなりません．中でどのような優先順位で処理がなされ，どれくらいのオーバヘッドの処理時間が必要なのかが把握できないOSを使った場合は，思わぬトラブルが発生するかもしれません．

筆者の経験上，もっとも難しく注意深く設計しなければならないのは，シンボル同期の部分です．受信される信号のS/N比はさまざまで，しかも，いろいろなレベルのフェージングの影響を受けています．あらかじめMATLABのようなシミュレーションソフトを使って，事前に検討しておくことも必要になります．また，実際のさまざまな悪条件の中でテストを繰り返すことも必要です．ここの部分は，教科書的な回答はありません．問題が発生するたびに解決しなければならないノウハウの部分です．

周波数選択性のフェージングに対応するためには，エラー訂正符号を使って対処しなければなりません．このモデムの場合も，ブロック符号によるエラー訂正符号が組み込まれています．さらに周期的なレベルフェージングに対応するためには**図58**のようなインタリーバを使ってバースト状の誤りを全体に拡散してエラー訂正コードで訂正できるようにしてやる必要があります．コーディングの部分は今回の記事の範疇外なので詳しい説明は省略します．しかし，実際の応用ではとても重要ですし，必ず埋め込まなければならない技術なので，しっかり理解しておく必要があります．

参考・引用文献

1) Richard Van Nee, Ramjee Prasad, *OFDM FOR WIRELESS MULTIMEDIA COMMUNICATION*, Artech House Publishing, 2000
2) John B. Anderson, *Digital Transmisson Engineering*, IEEE Press, 1999
3) ARIB STD-B13：『800MHz帯OFDM変調方式』，(社)電波産業会，2000
4) 宮津和弘，『Bluetoothガイドブック』，日刊工業新聞社，2000
5) 西村芳一，「SSB変調方式とDSP処理」，『HamJournal』，No68，CQ出版(株)，1990
6) 西村芳一，『DSP処理のノウハウ』，CQ出版(株)，2000
7) 西村芳一，『無線によるデータ変復調技術』，CQ出版(株)，2002

にしむら・よしかず　(株)エーオーアール

広い帯域を利用し，100Mbps以上の伝送を実現する

Chapter 4

60GHz帯を使った高速無線伝送技術

● 荘司洋三/辻　宏之

無線LAN規格の代表的なものの一つは2.4GHz帯の電波を使用するIEEE802.11bで，カタログなどでは最高伝送速度11Mbpsがうたわれている．また，5GHz帯の電波を使うIEEE802.11a（～54Mbps）も対応製品が出はじめた．一方，100Mbps以上のデータ伝送を実現する，60GHz帯の電波を使った無線伝送も研究が進められている．本章では，この60GHz帯を使った無線伝送技術の基礎と，課題である周波数安定性の問題を解決するための「ミリ波自己ヘテロダイン伝送方式」について解説する．（編集部）

はじめに

2.4GHz帯を使用した無線LANは手軽に利用できることから，その利用範囲がますます広がっています．また，5GHz帯の周波数の利用も認められるようになり，さらなる無線LANの発展が期待されています．現在もっとも利用者が多い2.4GHz帯の無線LANは，**ISM（産業科学医療用）バンド**と呼ばれる免許不要バンドを使用しており，5GHz帯の無線LANもやはり免許不要バンドを使用しています．このように，免許不要のバンドは一般ユーザーへの無線システムには魅力的なバンドといえます．現在これら2GHz帯や5GHz帯では，それぞれでおよそ100MHz幅程度の帯域の利用が可能となっています．

しかし，2.4GHz帯のISMバンドでは，本来の用途に加えて，無線LANや電子レンジなどを含むさまざまなシステムが利用するようになってきています．このため，同じ周波数帯を使う異なるシステム間の干渉が深刻な問題になりつつあります．また，同一システムにおいても，利用可能な周波数帯域が制限されているため，伝送速度は現在では数十Mbpsが最高となっていると同時に，利用するユーザー数が増えるにつれて伝送速度が低下するなどの問題点があります．

一方，近年高品位な動画像の無線伝送や光ファイバを用いたデータリンクの無線リンクへの置き換えなど，100Mbps以上の伝送速度が必要とされる無線アプリケーションがあり，さらに高速な無線伝送システムが求められています．100Mbps程度以上の伝送を実現する方法として，いくつかのシステムがこれまでに提案されています．

そのなかに，最近注目されている**UWB（Ultra Wideband）伝送方式**があります．これは既存の無線伝送システムと周波数帯を共用しつつも，比帯域で25％以上という従来の無線伝送システムと比較してきわめて広い周波数帯域を使用することで，帯域あたりの空中線電力を充分に低くおさえ，既存システムに干渉を与えることなく高速無線伝送を実現する手段です．

このように使用する帯域を広げることは高速伝送を実現する一つの方法ですが，**マイクロ波帯**で新たに広い帯域を確保することは一般に難しく，100Mbps以上の伝送を実現するためには，他のシステムと周波数を共有しなくてはいけません．これに対し，今回ここで取り上げるミリ波帯（30～300GHz，波長1～10mmの電波）は他のシステムと干渉することなく，広い帯域が利用できるという理由で注目されています．とくに60GHzミリ波帯は後で述べるように，7GHzという広い帯域がすでに免許を必要とせず利用可能な状況にあります．

本章では，ミリ波帯の60GHz帯をとりあげ，その特徴や標準化などについて述べた後，ミリ波帯における高速伝送を低コストに実現する技術や，これを利用したシステム開発例を紹介します．

1 ミリ波帯とは？

1.1 ミリ波帯の特徴

ミリ波帯とは，周波数30～300GHz（波長10～1mm）の電波を一般にさします．これまで60GHz帯のような高い周波数帯は，取り扱いが難しいなどの理由から一般の通信にはあまり利用されませんでしたが，近年のマイクロ波帯での周波数不足と**MMIC技術**（コラム参照）など回路技術の発達によって，その利用が注目されるようになりました．一般にミリ波帯の電波を利用する無線伝送システムのおもな特徴としては，

- 広帯域伝送が可能
- 機器の小型化が可能
- 伝搬による減衰が大きい

などがあげられます．ここでは60GHz帯に注目して，その特徴や利用状況を説明します．

● ミリ波の伝搬特性

一般に，電波の有効受信電力P_rは送信点と受信点間の距離をd，波長をλとした場合，

コラム 本章中に出てくる用語の説明

【ISMバンド】

ISM(Industrial, Scientific and Medical Band)(産業科学医療用)バンド．電子レンジや医療用加熱装置など，電波のエネルギを直接利用する特別な装置のために割り当てられた無線周波数帯．通信に使う場合は無線免許なしで自由に使えるが，通信品質の保証はない．

【UWB(Ultra Wideband)伝送方式】

無線通信の方式の一つで，データを1GHz程度のきわめて広い周波数帯に拡散して送受信を行うもの．それぞれの周波数帯に送信されるデータはノイズ程度の強さしかないため，同じ周波数帯を使う無線機器と混信することがなく，消費電力も少ない．UWBは位置測定，レーダ，無線通信の三つの機能を合わせもっており，きわめて独特な無線応用技術といえる．

【マイクロ波帯】

およそ1GHz(波長30cm)～30GHz(波長10mm)の電磁波．UHF波との境界域(1～3GHz)を準マイクロ波と呼び，ミリ波との境界域(20～30GHz)を準ミリ波と呼ぶこともある．

【MMIC】

Monolithic Microwave Integrated Circuit の略称．高周波(マイクロ波)帯で動作するガリウム砒素，あるいはシリコン集積回路(IC)を指す．携帯電話などで電波の送受信に多く使われている．

【加入者系無線アクセスシステム】

加入者系無線アクセスシステムは，WLL (Wireless Local Loop)やLMDS(Local Multipoint Distribution Service)など，さまざまな名称で呼ばれてきたが，ITUの1997年世界無線通信会議で，Fixed Wireless Access(FWA)を統一用語として採用することになった．

【多相位相変調方式】

多相PSK(Phase Shift Keying)変調方式．かぎられた周波数帯域で比較的多くの情報を伝送することのできる変調方式の一つ．多値数としては4相，8相のものが主として使用されている．とくに4相PSKのことをQPSK(Quadrature Phase Shift Keying)変調と呼ぶ．

【多値直交振幅変調変調方式】

多値QAM(Quadrature Amplitude Modulation)変調方式．かぎられた周波数帯域でもっとも多くの情報を伝送することのできる変調方式の一つ．多値数としては16値，64値などがおもに使用されており，最近では256QAMや1024QAMなども開発されている．しかし増幅器などによる非線形歪に弱いことから，64値以上はあまり用いられていない．

【OFDM方式】

Orthogonal Frequency Division Multiplexing(直交周波数分割多重)方式．周波数多重方式の一つで，多重するキャリアを密に配置できるため伝送効率が高い．5GHz帯を用いた無線LANで使用されているほか，ディジタル地上波放送への適用が決まっている．

【100Base-TX】

IEEE802.3が規定する非シールドより対線(UTP)を使う100Mbpsイーサネットの物理層仕様の一つ．名称の最後の“X”は，使用するより対線の仕様がANSI X3T9.5分科会が規定したFDDI/CDDIを基にしていることを示す．使用するケーブルは，カテゴリ5の2対(4芯)UTPである．

【マルチパス】

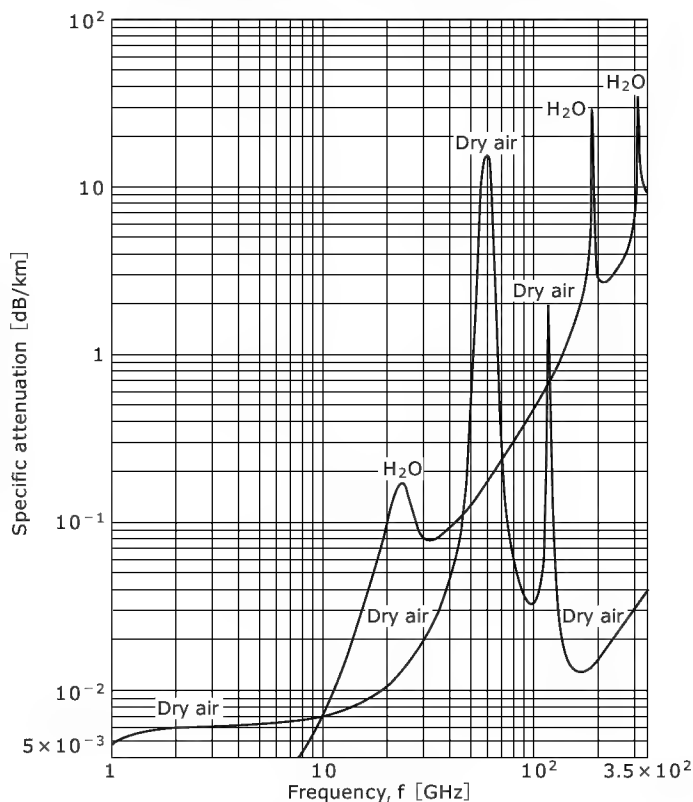
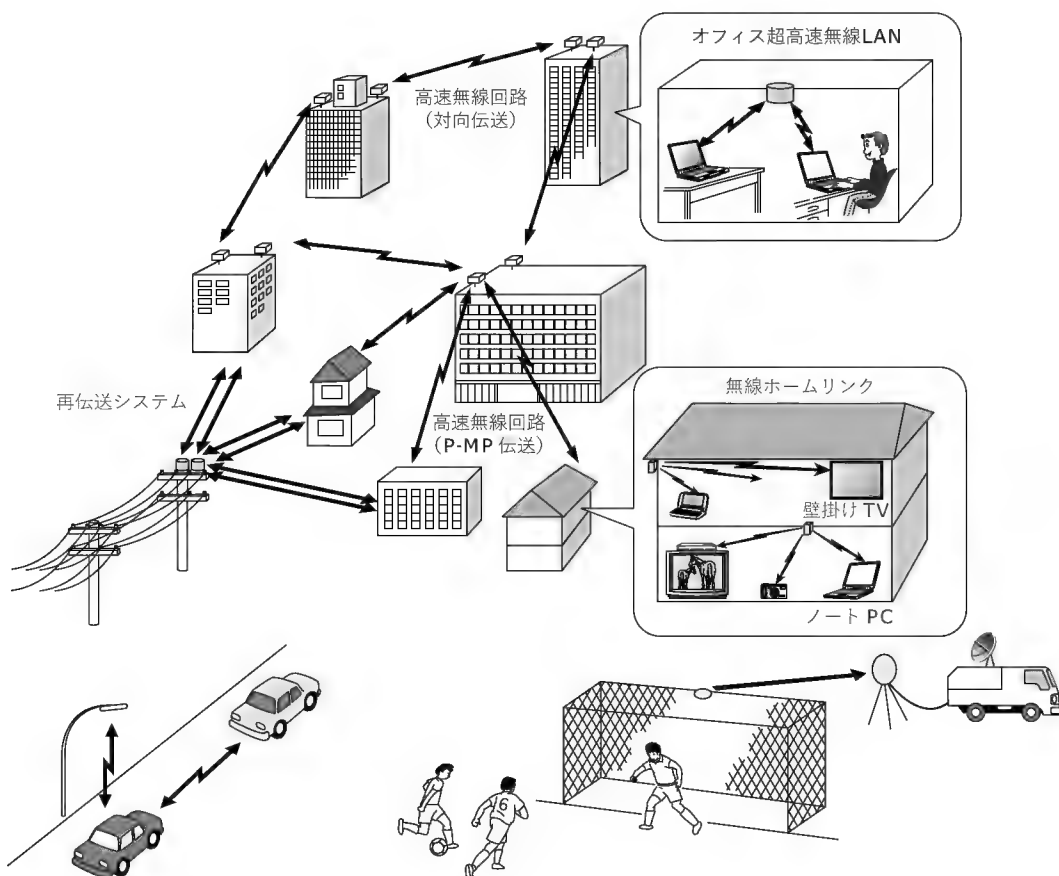
一般に，電波が複数の経路をたどって受信点に到達することをいう．これによって直接届く波(直接波)と遅延して届く波(遅延波)が干渉してデータ誤りの原因となる．地上ディジタル放送波に使用される予定のOFDM変調方式は，このマルチパス妨害に強い方式として知られている．

$$P_r \propto \left(\frac{\lambda}{d}\right)^2$$

(∝は比例の意)

の関係をもちます．この式より，60GHz帯のように波長が短い電波ほど，伝搬するにともなって急激に受信電力が減衰することがわかります．さらにマイクロ波帯と比較して，ミリ波帯はとくに降雨や大気による減衰が大きいことが知られています．**図1**に，電波の大気による吸収特性を示します．この図からもわかるように，とくに60GHz帯は大気(酸素)による電波の吸収減衰が大きい周波数帯であることがわかります(およそ16dB/km)．このような伝搬環境のなかで，しかも後述するように免許不要となる小電力(日本では10mW以下)の空中線で通信を行う必要性を考慮すると，おのずから60GHz帯の電波を用いる無線伝送システムは，あまり長距離伝送には向かないと考えられます．これは，無線伝送システムとしては欠点のように思われるかもしれませんが，ごく狭い閉じたエリア内のみに展開できる無線システムを容易に構築できることから，周波数の空間的な繰り返し利用が可能になり，周波数を有効に利用できるという利点ととらえることもできます．

さらに，ミリ波帯は非常に周波数が高いことからマイクロ波帯以下の電波と赤外線など光波の中間的性質をもちます．つまり光波のように非常に直進性が強いことから，回折波や反射波などは利用せず直接波のみを受信するいわゆる“見通し内(LOS：Line of sight)通信”を行うことを前提にシステムが設計される一方で，ふすま・障子やガラス，屋内で使用される薄い内壁材などは透過します．このような特徴は，人物やコンクリート壁などで見通しを遮られると通信が不可能になるなどの問題を生じさせますが(後述)，通信者が意図しない場所で信号を傍受される可能性がきわめて低いため，高いリンクセキュリティが実現できるという利点があります．つまり，現在の無線LANで問題となっているようなセキュリティ対策をプロトコル上で強化する必要はあまりなく，ハードウェアに対する負担を軽減できる

〔図1〕電波の大気による吸収特性
(ITU-R Recommendations, Vol.1997, Pseries, Part1より)

〔表1〕60GHz帯の免許を要しない無線設備のおもな技術的条件

通信方式	単行通信方式，単信方式，半複信方式，複信方式および同報通信方式
変調方式	規定しない
空中線電力	10mW以下
周波数の許容偏差	$\pm 500 \times 10^{-6}$以下
占有周波数帯域	2.5GHz以下
送信機空中線利得	47dBi以下

と考えられます．

1.2. 国内外における60GHzミリ波帯の利用状況

● 日本

ミリ波帯の中では，これまでに30GHz帯で防災用通信や**加入者系無線アクセスシステム**での利用が実用化されてきました．60GHz帯については，2000年に電波法が改正され，59GHzから66GHzまでの7GHzが免許不要バンドとして使用できるようになりました．**表1**にそのおもな技術的条件を示します．このように60GHz帯を用いる免許不要バンドは，マイクロ波帯における免許不要バンドと比較してきわめて広い周波数帯域が利用可能となっており，また**表1**からもわかるように，通信方式や変調方式，周波数の許容偏差などに対する規定がゆるいことがわかります．このように規定することで，システムの導入の促進

〔図2〕60GHz帯を使用する無線システムの利用例

〔表2〕60GHz帯において利用が想定される無線システム

システム名	利用形態	伝送内容
映像多重伝送システム	おもに屋内において，地上波，BS，CS放送およびケーブルTVなどの映像情報をアンテナ線不要で受像機に伝送する	地上テレビ放送，BS，CS，CA放送の信号
超高速無線LAN	無線化したLANによる高速伝送を行う．また，有線で接続されていたLANの無線化	
無線ホームリンク	家庭やSOHO内の電子機器の無線による接続	マルチメディア情報
移動体アクセスシステム	停車中または移動中の車両に対する大容量のマルチメディア情報の伝送を行う	マルチメディア情報(地図情報，交通情報，地域情報など)
衛星放送番組再伝送システム	立地条件によりBSやCSを直接受信できない集合住宅などに対して再放送を行う	BS，CS放送
画像伝送システム	スポーツ中継などにおいて，カメラマンが立ち入れない場所や移動体などからの映像を中継車まで伝送	HDTV素材信号
車車間通信システム	急ブレーキなどの危険情報を車両間に伝送	危険情報など
電気通信業務用高速無線回線システム	将来のマルチメディア伝送に対応した大容量無線中継回線に使用．ビル間通信や加入者宅へのアクセス，または光ファイバの補完用として使用	加入者無線サービス信号
ケーブルテレビ無線伝送システム	河川・鉄道横断などの立地条件により，ケーブルを敷設することが困難な場合において，映像信号などを伝送するシステム	多チャネル映像信号，データ，電話

をはかっています．その他の詳細については参考文献1)を参照してください．

上述したような特徴と技術的条件から，免許不要バンドとしての60GHz帯は伝送距離においては短距離であり，非常に高速大容量の無線伝送に適した周波数帯といえます．**図2**に60GHz帯を使用する無線システムの利用例を示します．家庭，オフィスなどの屋内では，超高速無線LANや無線ホームリンクなどの利用，また屋外ではビル間の高速通信や車車間通信，車載衝突防止レーダなどがおもに検討されています．また，今後利用が想定されているシステムを**表2**に示します．

● 欧米

米国では，当初1994年に59GHzから64GHzの5GHzが免許不要バンドとして割り当てられましたが，その後57GHz～59GHz帯が追加され，現在では57GHzから64GHzまでの7GHzが使用できるようになりました．規定では全電力が500mW以下であり，3m離れた地点で電力密度が9μW/cm^2(平均)，18μW/cm^2(ピーク)以下となっています[2)～5)]．すでに商品化されたものもあります(**表3**)．たとえばGigalinkはpoint-to-point通信で，最大622Mbpsの通信が可能であり，光ファイバの代用としておもに利用されます(**図3**)．

一方，ヨーロッパでは54.25～66GHz帯の使用が検討されていて，57～59GHzを免許不要バンド，59～62GHzを無線LANに使用する国が多いようです．

2 60GHz帯無線伝送システム開発における基本的な課題

前述したように国内における電波法上では，他の免許不要バンドと比較して60GHz帯は非常に広帯域な周波数を利用できるという利点をもちます．また，装置で使用されるデバイスやアンテナなどは一般に使用する波長によってそのサイズが決定付けられるため，60GHz帯のように非常に短い波長を用いる無線伝送システムの場合，装置を非常に小型化することが容易になります．このような利点が存在する一方で，60GHz帯の無線システムは，従来のマイクロ波帯の電波を使用するシステムと比較すると，その周波数の高さに起因したいくつか取り扱いにくい点や問題が存在します．以下では，そのような問題点について説明します．

〔表3〕欧米における60GHz帯の製品の例

製造会社	Terabeam[6)]	Nokia[7)]
製品名	Gigalink	Nokia MetroHopper Radio
用　途	屋外ポイント-ポイント	携帯電話アクセス用
使用可能距離	～1.25km	−
伝送速度	100～622Mbps	2Mbps × 4
その他	インターフェース： Ethernet(100Mbps)， OC-3/STM-1(155Mbps) OC-12/STM-4(622Mbps)	変調方式：MSK 出力：5dBm

〔図3〕Gigalinkの外観

2.1 デバイス開発面での課題

近年のMMIC技術，およびその他の平面回路実装技術の発達にともなって，60GHz帯のような非常に高い周波数帯おいても，増幅器，ミキサ，発振器，周波数逓倍器などのアクティブデバイス，およびフィルタ，アンテナなどのパッシブデバイスを微小面積の平面回路上に実装して，RFセクションに必要な機能全体を非常に小型なRFモジュールへ収めることが技術的に可能になっています[8)]．開発例として，**図4**にMMIC技術によって開発された60GHz帯無線伝送システム用送受信RFモジュールを示します．このモジュールのサイズはおよそ50 × 23 × 6mmで，

無線装置のRFセクションとして必要な機能である，RFアンプ，RFバンドパスフィルタ（BPF），局部発振器（ローカル発振器），周波数逓倍器，ミキサ，アンテナ（送信用のみ）のすべてが納められています．以下，デバイスの機能別に，システム設計に関係する開発課題について簡単にふれておきます．

① RFアンプ

送信回路には送信信号を必要な空中線電力まで増幅するためのパワーアンプ（PA）が必要となります．したがってPAの開発においては第一に高出力化が重要となりますが，出力信号に歪みが生じるとデータ誤りの原因となるため，同時に十分な線形性（リニアリティ）も満足することが重要な開発課題となります．これは，一般にはPAの飽和レベルからバックオフさせたレベルで使用することで満足されます．このことから，日本では60GHz帯の免許不要バンドで使用できる空中線電力の上限は10mWですが，実際にはこれ以上の出力が可能なPAの開発が必要であることがわかります．

一方の受信回路には，受信信号を復調可能なレベルまで増幅するための低雑音アンプ（LNA）が必要となります．信号レベルは受信時には非常に微弱であるため，これを後の復調回路で扱うことが可能なレベルまで増幅しますが，このとき所望の信号が増幅時に発生する雑音で埋もれてしまうことのないように，非常に低雑音なものが求められます．一般には，受信回路の初段で使用されるLNAの雑音性能（NF）で受信回路の受信感度が決定されます．

幸いなことに，60GHz帯のシステムにおいて上記のようなPAやLNAはすでにGaAs（ガリウム・ヒ素）材料をベースとしたMMICで実用レベルにいたっています．

② RFバンドパスフィルタ（BPF）

一般的な高周波帯の無線伝送システムや60GHz帯無線伝送システムの送信回路では，ローカル信号とミキサで中間周波数帯（IF）信号を60GHz帯へ周波数変換した後に生じるイメージ成分と，ローカル信号成分を，このBPFを使用して除去します．一般に作成可能なBPFの通過帯域幅（パスバンド）や実現可能な周波数特性の急峻性の程度は，比帯域すなわち（通過帯域幅/中心周波数）で決まるため，同じ通過帯域幅でも60GHz帯のように高周波になるほどその作成は困難になります．そこで一般には，IF周波数を高くしてRF帯で分離する信号の周波数間隔を大きくし，BPFの負担を減少させるなどの工夫をしています．また，受信回路においても不要な帯域外の干渉信号などが入力されて，後の回路で所望の信号に悪影響を及ぼさないように，信号入力部にもBPFが挿入されます．開発課題としては，挿入損失の低減，急峻な周波数特性，およびパスバンド内での平坦な周波数特性の実現などがあります．

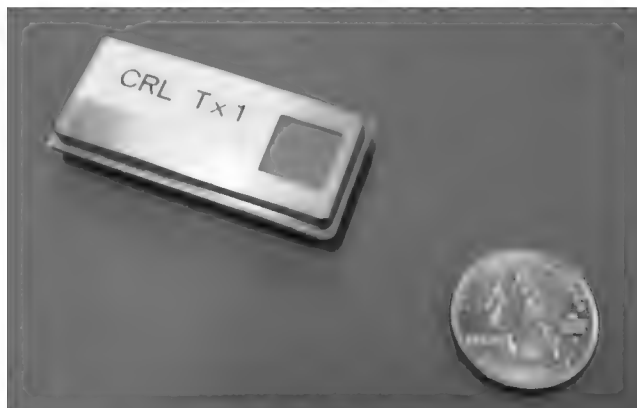
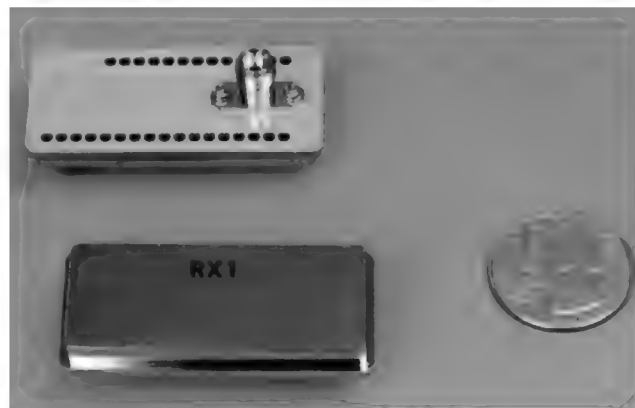
③ ミキサ

ミキサは，送信回路ではIF信号を60GHz帯へアップコンバートし，逆に受信回路では60GHz帯の受信信号をIF帯へダウンコンバートするデバイスで，ローカル発振器とともに使用されます．コンバージョンロス（挿入損失）の低減以外に，60GHz帯では比較的広帯域な信号をまとめて周波数変換するため，使用帯域内での周波数特性の平坦化はやはり重要です．60GHz帯では，アンプ回路同様GaAsをベースとしたMMICで実現可能となっています．

④ 局部発振器（ローカル発振器）と周波数逓倍回路

送受信回路での周波数変換において，ミキサとともに必要不可欠なデバイスがローカル発振器です．しかしながら，ミキサへ入力される60GHz帯のローカル信号を直接生成することは通常困難であるため，所望周波数の2分の1もしくは4分の1の発振周波数をもつ発振器出力を周波数逓倍回路で2逓倍もしくは4逓倍する構成を用いることが多いようです．60GHz帯のような非常に高い周波数帯では，未だ周波数的に高安定な発振器の実現が技術的に困難な状況にあります．

上記で紹介したような小型RFモジュールへローカル発振器を内蔵する場合には，比較的低コストで小型化が可能な誘電体共振型発振器（DRO：Dielectric Resonated Oscillator）を用いる場合があります．しかしながら，このような小型な発振器は周波数安定性が良くないなどの問題もあり，システム性能に重大

〔図4〕MMIC技術を用いて開発したミリ波帯RFモジュールの例

（a）送信モジュール

（b）受信モジュール

な影響を与えます．この問題については，後でより詳細に取り上げることにします．他方の周波数逓倍回路については，やはりアンプなどと同様にMMICで実現可能となっています．

⑤ **アンテナ**

60GHz帯のシステムで利用可能なアンテナの種類については，マイクロ波帯のシステムと同様のものがすべて利用可能ですが，前述したようにそのサイズは波長によって決まるため，小型化が可能です．とくに高いアンテナゲインを必要としない場合には，非常に小型(約2.5mm平方)のパッチアンテナを他の回路とあわせて，**図4**のようなRFモジュールに内蔵することが可能になります．アンテナの種類についてはパッチアンテナのほかに，スロットアンテナや，導波管アンテナ，ホーンアンテナなどが使われ，それぞれ一長一短の特徴があります．さらに高いアンテナゲインが必要な場合には，それらを複数配置するアレイ構成が用いられます．

3 発振器の安定性について

3.1 周波数ドリフト・位相雑音と評価方法

前述したデバイス開発における課題のうち，ここではとくに60GHz帯の無線伝送システムの性能と製造コストに大きく影響を与える，ミリ波帯ローカル発振器の安定性問題とその評価方法について取り上げます．

通常，発振器は誘電体などの材料を使用して，その材料とサイズによって固有となる共振周波数の原理によって発振周波数を生成しています．したがって，デバイスの温度上昇などによって材料が微少に膨張すると，それに応じて微少に発振周波数が変化してしまいます．このようなおもにデバイス温度などによって周波数が非常にゆっくりと漂動することを**周波数ドリフト**と呼び，その性能は(ppm/℃)の単位で評価されます．

一方，実際の発振器から得られる発振信号は，かりにその周波数が平均的には常に一定だとしても，理想的な三角関数波形である正弦波の信号と比較すると，その位相がランダムに進んだり遅れます．このような成分を雑音とみなして，位相雑音と呼びます．位相雑音を評価する場合には，一般に振幅一定の無変調キャリアを前提として，総キャリア電力に対するキャリアの中心周波数からオフセットしたある周波数点における1Hz帯域あたりの雑音電力の比(dBc/Hz@オフセット周波数)が使用されます．

ところでミリ波帯のローカル信号を得るには，通常より低い発振器の信号を周波数逓倍回路で逓倍して得ることが多いことをすでに述べましたが，上述した周波数ドリフトや位相雑音は周波数逓倍回路の使用によってさらに悪化することが知られています．これは，Δfの周波数ドリフトが発生しているキャリアを2逓倍した場合の周波数ドリフト量が$2\Delta f$となることからも容易に説明がつきます．また位相雑音電力については，2逓倍回路によって6dB劣化することが知られています．

3.2 周波数の不安定性と変調信号

上述した周波数ドリフトや位相雑音のような周波数の不安定性が無線伝送システムの性能に与える影響は，使用する変調方式によって異なります．たとえば，BS放送のアナログチャネルやラジオ放送などで現在も使用されているアナログFM変調方式は，相対的なキャリア周波数の変動に信号が重畳されているため，多少の周波数ドリフトや位相雑音の影響はあまり受けません．しかしながら，近年このようなアナログ変調方式は使用されなくなる傾向にあります．

一方，雑音による信号品質劣化の改善や周波数利用効率の改善が容易なことから，**多相位相変調(M-array PSK : Phase Shif Keying)方式**や**多値直交振幅変調(M-array QAM : Quadrature Amplitude Modulation)方式**などのディジタル変調方式が多用される傾向にあります．このようなディジタル変調方式はいずれもキャリアの位相状態に情報を重畳するため，その伝送システムで使用する発振器には非常に優れた位相雑音特性が要求されます．なかでも5GHz帯を使用する無線LAN(IEEE802.11aやHiperLAN2，HiSWANなど)でも標準となっている**OFDM(Orthogonal Frequency Division Multiplexing)方式**は，周波数利用効率の向上とマルチパス伝搬環境による信号品質劣化の対策に有効であるため，最近注目されていますが，この信号を伝送する場合には，さらに周波数安定性に対する要求値が厳しくなることが知られています．

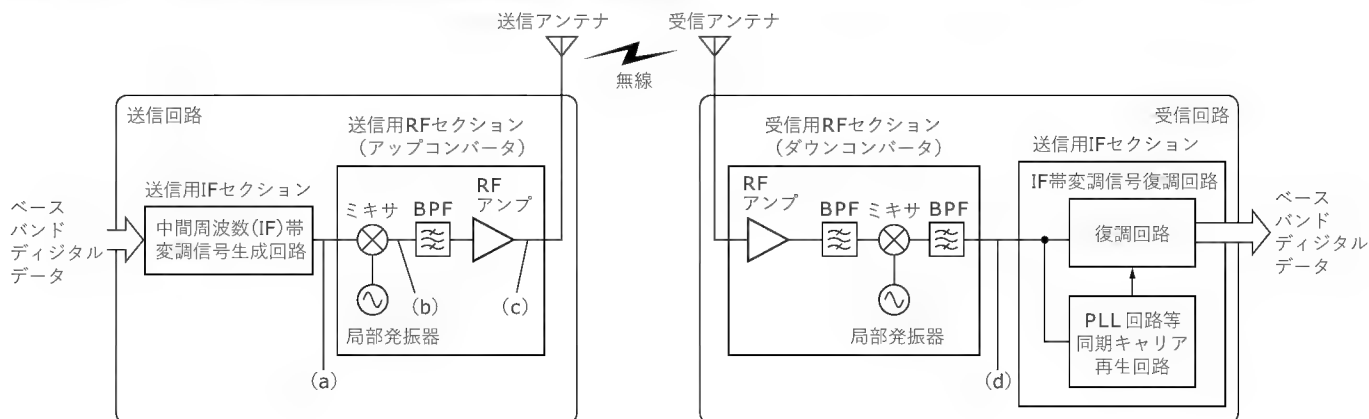
4 スーパーヘテロダイン伝送方式の原理と問題

本節では，高周波を用いる無線伝送システムにおいて従来から一般的に使用されてきた，スーパーヘテロダイン伝送方式の原理とその問題点について解説します．

4.1 スーパーヘテロダイン伝送方式のシステム構成

図5に，スーパーヘテロダイン伝送方式を用いる無線伝送システムの構成を示します．この図からもわかるように，高周波を使用する無線伝送システムの送信回路では，まず比較的低い中間周波数(IF)帯で変調信号を生成した後，これを送信用RFセクションにおいて無線周波数帯へ周波数変換(アップコンバート)してアンテナより送信します．逆に受信回路では，アンテナより受信した信号をまず受信用RFセクションで中間周波数帯へ周波数変換(ダウンコンバート)した後，これをIF帯復調回路において復調します．

また，IF帯復調回路においては通常，同期検波のためのPLL回路やコスタスループ回路などが組まれていて，受信IF信号から信号の復調に必要な同期キャリアの生成を行います．ここで注目する点は，送信用および受信用のRFセクション両方においてIF-RF間の周波数変換を行うためにローカル発振器が必要となることです．ここで必要となる具体的な周波数の例をあげると，たとえば中心RF周波数f_{RF}が60GHzの無線伝送システムを設計する場合，1GHzのIF中心周波数を用いると仮定すると，

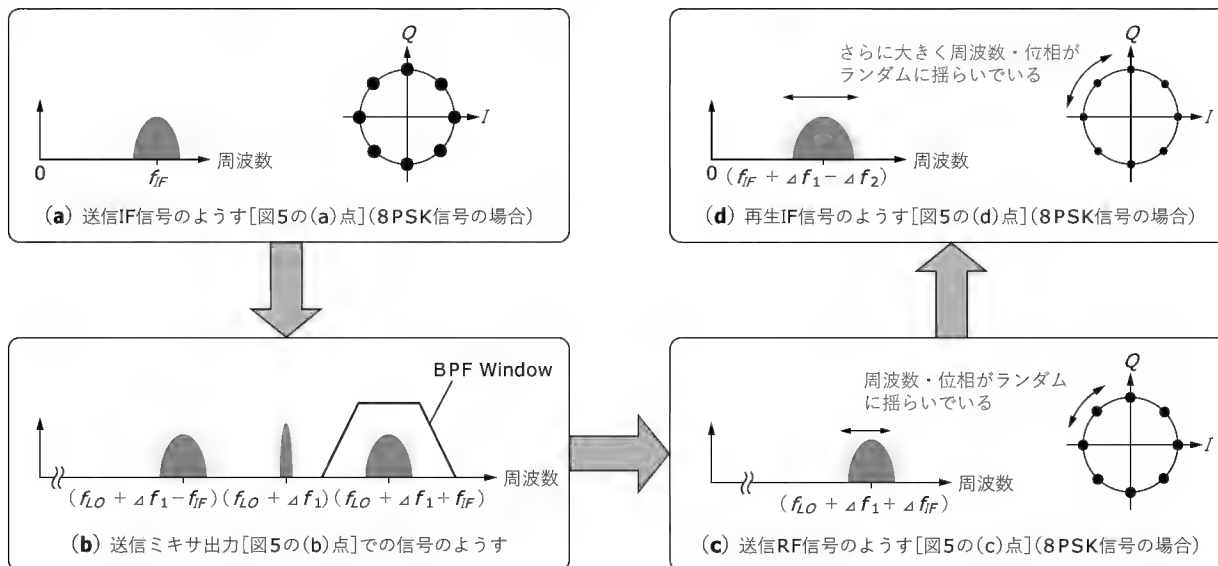
〔図5〕スーパーヘテロダイン伝送方式を用いる無線伝送システムの構成

RFセクションで必要となるローカル発振器の発振周波数は送受信回路ともに59GHzとなります．

4.2 発振器の不安定性がシステムに与える影響

60GHz帯では発振器について十分な周波数安定性を得ることが未だ困難であることをすでに述べましたが，送信機で使用するローカル発振器が周波数的に不安定であった場合，その周波数不安定性は得られるRF信号に直接重畳されることになります．さらに受信機では，もう一度使用されるローカル発振器の周波数不安定性がダウンコンバージョン時に信号に重畳されます．

すなわち，最終的には送信RFセクションに入力されるIF信号に，送信機と受信機両方のRFセクションで使用したローカル発振器の周波数不安定性があわせて重畳されたIF信号が受信機のRFセクション出力として得られることになります．以降では，このようすをスーパーヘテロダイン伝送方式のシステム構成図(**図5**)と，同図における各ポイント(**a**)～(**d**)点での信号スペクトルのようすを示した**図6**を参照しながら詳細に解説します．

送信回路におけるIFセクションの出力〔(**a**)点〕はベースバンド情報でディジタル変調された中心周波数f_{IF}の信号であり，その信号点配置はたとえば8相PSK変調であれば，**図6**(**a**)に示すような直交座標上の円周上を8で等分割した点のいずれかを離散的に取ります．このIF信号をRFセクションでは発振周波数f_{LO}のローカル信号で60GHz帯にアップコンバートしますが，このローカル発振器として小型かつ低コストなものを使用した場合，上記発振周波数f_{LO}に周波数ドリフト成分Δf_1が足しあわされて実際には周波数$(f_{LO}+\Delta f_1)$でアップコンバートされることになります．このようなアップコンバージョンによってミキサ出力〔(b)点〕に現れる信号はアップコンバージョンされた中心周波数$(f_{LO}+\Delta f_1+f_{IF})$の信号，イメージ成分である中心周波数$(f_{LO}+\Delta f_1-f_{IF})$の信号，そして中心周波数$f_{LO}$のローカル信号成分からなります〔**図6**(**b**)〕．通常はイメージ成分とローカル信号成分は不要な信号成分(スプリアス)として取り扱われるため，その後のBPFで所望の信号成分のみが取り出されます．こ

〔図6〕スーパーヘテロダイン伝送方式システムにおける各点での信号スペクトルのようす

のようにして最終的には〔(c)点〕で中心周波数($f_{LO}+\Delta f_1+f_{IF}$)のRF信号のみが抽出されて送信アンテナより送信されます．このように，この送信RF信号には送信用ローカル信号に起因した周波数ドリフト成分Δf_1が重畳されていることがわかります〔**図6**(**c**)〕．

一方，受信回路は受信した中心周波数($f_{LO}+\Delta f_1+f_{IF}$)のRF信号成分を，再び発振周波数f_{LO}のローカル信号を使用してIF帯にダウンコンバートしますが，やはりこの発振周波数f_{LO}には周波数ドリフト成分Δf_2が発生しており，実際には周波数($f_{LO}+\Delta f_2$)でダウンコンバートすることになります．その結果，〔(d)点〕で得られるIF帯変調信号は中心周波数が$(f_{LO}+\Delta f_1+f_{IF})-(f_{LO}+\Delta f_2)=(f_{IF}+\Delta f_1-\Delta f_2)$の信号となって，送信側の周波数ドリフト成分と受信側の周波数ドリフト成分の両方が重畳された信号となってしまいます．送信側で発生する周波数ドリフト成分Δf_1と受信側で発生する周波数ドリフト成分Δf_2は互いに独立に発生するため，結果的に非常に大きな周波数ドリフトが発生したIF信号が受信回路におけるIFセクションに入力されることになります．

以上では周波数ドリフトを例に説明しましたが，位相雑音についても位相の時間微分成分が周波数であることを考慮すると比較的高速かつランダムにゆらぐ周波数ドリフトと捕らえることが可能なので，同様に考えて非常に大きな位相雑音が発生したIF信号が受信回路におけるIFセクションに入力されることになります．

4.3 従来の発振器の不安定性問題に対する解決策

マイクロ波帯を用いる一般的な無線伝送システムにおいても，やはり発振器の周波数安定性問題や伝搬環境などに起因して，受信する無線信号には周波数ドリフトと位相変動が生じています．したがって受信回路では，このような周波数不安定性をともなう受信変調信号を復調(同期検波)するために，これと同期した基準キャリアを生成する必要があります．これらは通常，復調回路にPLL(Phase lock loop)回路や，コスタスループ回路などを装備することで実現するしくみになっています．

そこで60GHz帯の無線伝送システムでも，IFセクションにおける復調回路において同様の同期キャリア再生回路を装備する解決方法が考えられます．しかしながら，先に述べたような低コストで小型なローカル発振器を使用した場合に復調回路に入力されるIF信号には，従来のマイクロ波帯のシステムでは想定されていないほど大きな周波数ドリフトや位相雑音が発生しており，現状の技術でこれに対して即時に応答することが可能な同期キャリア再生回路の実現は非常に困難な状態にあります．また，これら従来のマイクロ波帯のシステムで使用されている同期キャリア再生回路の適用が可能なレベルまでローカル発振器を安定化するためには，非常に高価で特殊な外部回路を備えた発振器構成が，各RFセクション内で必要となります．

これでは，装置の小型化が可能という60GHz帯無線伝送システムの大きな利点を損なう結果になるだけでなく，システムのコスト高の大きな要因になり，後で述べるような60GHz帯無線伝送システムの一般家電やオフィスへの適用が難しくなってしまいます．

5 ミリ波自己ヘテロダイン伝送方式

本節では，先に解説したスーパーヘテロダイン伝送方式を用いて60GHz帯のようなミリ波帯の無線伝送システムを実現する場合の周波数安定性の問題を解決するために，筆者が平成12年度に独自に考案・開発した，ミリ波自己ヘテロダイン伝送方式について解説します．

5.1 ミリ波自己ヘテロダイン伝送方式のシステム構成

ミリ波自己ヘテロダイン伝送方式は，特別な周波数安定化技術を使用しない簡単かつ低コストな装置構成で，従来のスーパーヘテロダイン伝送方式ではミリ波伝送が困難であったM-array PSK変調信号やM-array QAM変調信号，さらにはこれらの変調方式を1次変調として使用したOFDM信号などの高効率ディジタル変調信号をも高安定にミリ波伝送することが可能になる新しい無線伝送方式です[9)～11)]．

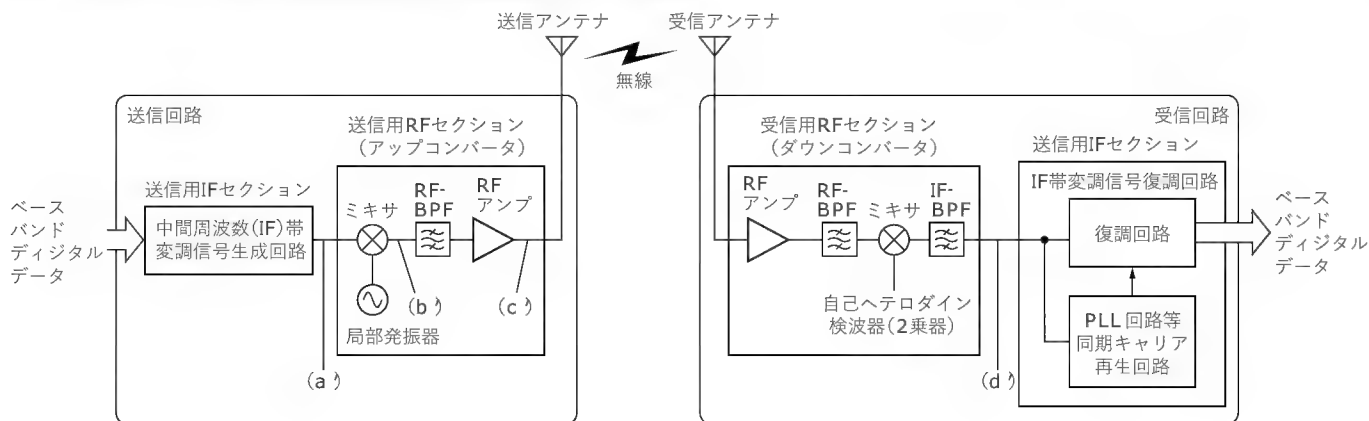
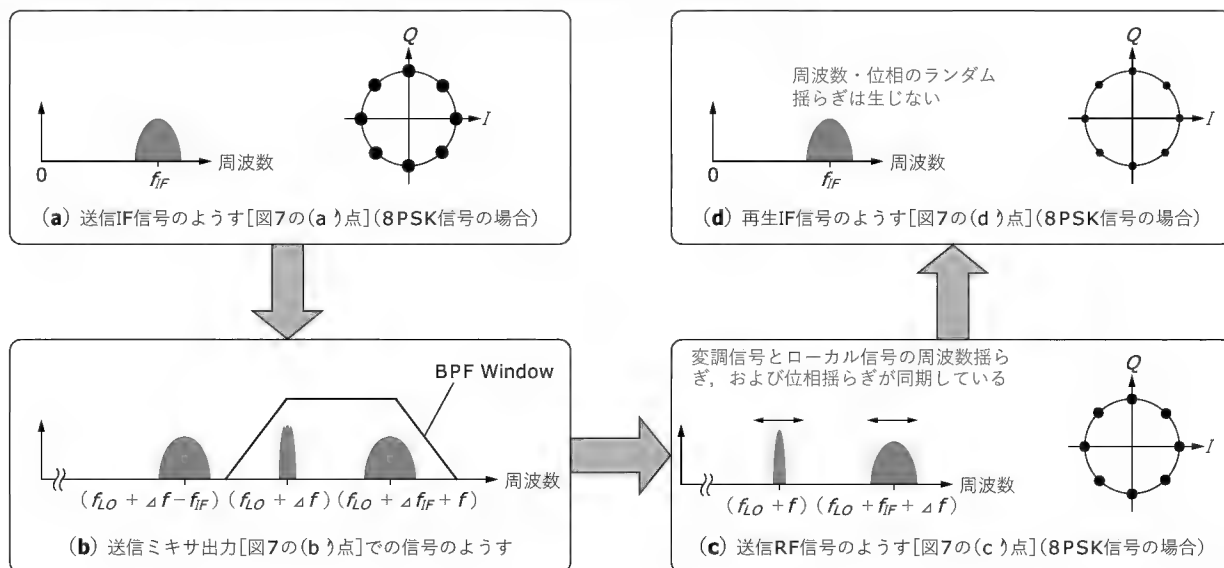
ミリ波自己ヘテロダイン伝送方式を用いた無線伝送システムの構成を**図7**に示します．ここで**図5**に示したスーパーヘテロダイン伝送方式を用いたものと比較してみると，異なる点はIF-RF間の周波数変換機能を担う送受信回路でのRFセクションのみであることがわかります．加えて，送信用RFセクションについてはその構成要素自体はまったく変わらないこともわかります．

送信回路におけるRFセクションは，やはりIF帯で生成された変調信号をRF帯へアップコンバージョンする機能をもちますが，このとき従来の伝送方式ではスプリアスとして充分に抑圧されてきたRF出力におけるローカル信号成分を，周波数変換されたRF変調信号とあわせて充分に出力されるよう，送信RFセクション回路内のBPFが設計してあります．通常送信用RFセクションにおけるミキサ回路の出力は，スーパーヘテロダイン方式での解説でも述べたように，アップコンバージョンされた上側帯波信号成分，イメージと呼ばれる下側帯波信号成分およびローカル信号成分からなりますが，それら出力成分から下側帯波成分のみをBPFを用いることで除去するわけです．

一方，受信回路におけるRFセクションではこれらの信号を受信アンテナで受信して増幅した後，2乗器として動作するミキサ回路で2乗検波(自己ヘテロダイン検波)することで，もとのIF帯信号へダウンコンバージョンを実現しています．これによって，受信回路ではコスト高の要因となるローカル発振器そのものが不要になると同時に，送信回路のRFセクションで使用したローカル信号の周波数ドリフトと位相雑音が完全にキャンセルされます．

5.2 周波数ドリフトおよび位相雑音キャンセルの原理

以下では，スーパーヘテロダイン伝送方式について説明したときと同様に，システム構成図(**図7**)における各ポイント(a)～

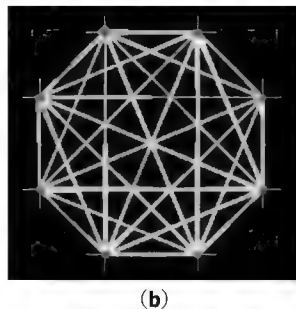
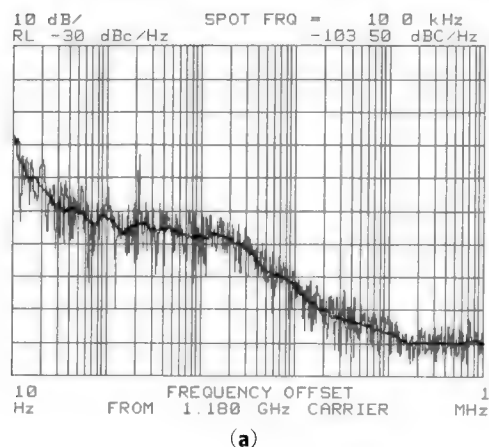
〔図7〕ミリ波自己ヘテロダイン伝送方式を用いる無線伝送システムの構成

〔図8〕ミリ波自己ヘテロダイン伝送方式システムにおける各点での信号スペクトルのようす

(d)点での信号スペクトルのようすを示した**図8**を参照しながら，周波数ドリフトおよび位相雑音が受信回路でキャンセルされる原理を解説していきます．

送信回路におけるIFセクションの出力は**図8(a)**のような伝送情報でディジタル変調された中心周波数f_{IF}，信号帯域Bの信号であり，その信号点配置は，たとえば8相PSK変調であれば，**図8(a)**に示すような信号の位相を示す円周上を8で等分割した点のいずれかを取ります．本信号がRFセクションでは発振周波数f_{LO}のローカル信号で60GHz帯にアップコンバートされますが，すでに述べたように本ローカル発振器として小型かつ低コストなものを使用した場合，上記発振周波数f_{LO}には周波数ドリフト成分Δfが発生しており，これが足しあわされて実際には周波数$(f_{LO}+\Delta f)$でアップコンバートされることになります．このようなアップコンバージョンによって得られるRF信号は中心周波数$(f_{IF}+f_{LO}+\Delta f)$，信号帯域Bの信号となり，局部発振信号と同等の周波数ドリフトが重畳されることになります．さらに，ミリ波自己ヘテロダイン伝送方式における送信回路はこのRF信号に中心周波数$(f_{LO}+\Delta f)$のローカル信号成分をあわせて送るため，最終的な〔(c)点〕での送信RF信号スペクトルは**図8(c)**のようになります．

一方，受信回路では中心周波数$(f_{IF}+f_{LO}+\Delta f)$のRF信号成分と中心周波数$(f_{LO}+\Delta f)$のローカル信号成分をまとめて受信，増幅した後，不要な信号成分がBPFで除去されて，2乗検波(自己ヘテロダイン検波)されます．これによって両成分の差周波数成分が発生することになり，これを所望の信号として取り出します．ここで注目することは，両成分に同期した周波数ドリフト成分Δfが等しく含まれていることであり，検波時に本周波数成分も同時に差し引かれる点です．すなわち，$(f_{IF}+f_{LO}+\Delta f)-(f_{LO}+\Delta f)=f_{IF}$となり，周波数ドリフト成分と位相雑音成分がキャンセルされて，送信用ローカル発振器の周波数不安定性の影響をまったく受けない中心周波数f_{IF}のIF帯変調信号を再度取り出すことが可能となります〔**図8(d)**〕．

〔図9〕送信用IFセクション出力信号の位相雑音特性と復調信号点配置

(a)

(b)

〔図10〕ミリ波帯局部発振信号の位相雑音特性

5.3 実験例の紹介

次に，筆者が行った実験結果をもとに，実際に本ミリ波自己ヘテロダイン伝送方式で信号を伝送した場合の，位相雑音特性および復調信号品質のようすを紹介します．本実験に使用したRFセクションは，先に**図4**に示した数センチ角のRFモジュールで構成されており，送信用については本モジュール内にアンテナを含むRFセクションとして必要な機能がすべて，受信用についてはアンテナを除く機能がすべて収められています．

〔図11〕受信用IFセクション入力信号の位相雑音特性と復調信号点配置

(a)

(b)

図9は，送信回路のIFセクション出力点における位相雑音特性と信号伝送実験に使用した8相PSK変調のIF信号を復調した場合の信号点配置のようす(シグナルコンスタレーション)を示したものです．**図9**に示した測定結果のグラフの横軸はIF中心周波数1.18GHzからのオフセット周波数(Hz)であり，縦軸は同オフセット周波数におけるキャリア電力に対する位相雑音電力の比の値(dBc/Hz)です．ここでは，グラフの横軸が対数となっていることに注意してください．同図からもわかるように，一般的な発振器ではオフセット周波数が大きくなるほど位相雑音電力が減少します．また，示したグラフからは10kHzオフセット周波数点で−103.50(dBc/Hz)の値が読み取れますが，〜数GHz程度のマイクロ波帯の発振器では，この程度の位相雑音特性を実現することが現状技術で充分可能です．

一方，**図10**はこのようなIF帯の信号を60GHz帯へアップコンバージョンするために，送信用RFセクション(送信RFモジュール)内で使用した中心周波数59.01GHzの局部発振信号の位相雑音特性を示しています．オフセット周波数に対する位相雑音電力の値が，**図9**に示したIF帯のものと比較してみると大幅に劣化していることがわかりますが，先に写真でも示した小型RFモジュール内に実装可能な60GHz帯発振器の実力値としては，いまだこの程度にあります．この側定例では，10kHzオフセット周波数点ではおよそ−49(dBc/Hz)の値が読み取れますが，同オフセット周波数点におけるIF帯のものと比較すると，じつに50dB以上の開きがあります．なお，これらのグラフは位相雑音電力の特性を示したものですが，実際にはさらに温度などの影響によって中心周波数がゆっくりと，しかしMHz/分単位の大きさで変化する周波数ドリフトが観測されます．

さて，送信RFセクションにこのようなローカル発振器を使用したミリ波自己ヘテロダイン伝送方式で，IF信号をRF伝送した後，受信回路内のRFセクションで検波した後に得られたIF信号の位相雑音特性と，得られた8相PSK変調信号を伝送して復調した場合の信号点配置のようすを**図11**に示しました．位相雑音特性については10kHzオフセット周波数点で−102.83(dBc/Hz)の値が読み取れますが，**図9**に示した送信IF信号における値からの劣化量は1dB未満であることがわかります．これはほとんど測定誤差の範囲内であり，オフセット周波数全体を見た場合でもまったくといってよいほど劣化していないことがわかります．また，8相PSK変調信号を復調した場合の信号点配置についても当然のことながら，送信回路におけるIF出力のものからほとんど劣化していないものが得られています．

6 屋内向けミリ波映像多重伝送システム

6.1 開発の背景

60GHz帯では空中線電力10mW以下という制約はあるものの，7GHzにわたる非常に広帯域な免許不要バンドの利用が可能であり，1無線局あたりの占有帯域幅は2.5GHzまでといった他のバンドには見られない広い帯域が使用できます．したがって，このような広帯域性を利用して1Gbpsを超える超高速な無線データ通信を実現することは原理的には可能です．しかしながら，現段階ではデバイスとシステム開発に要するコストが非常に高く，たとえ製品化したとしてもそれは非常に高価なものになってしまうため，実質的な普及を見込むことは困難に思えます．

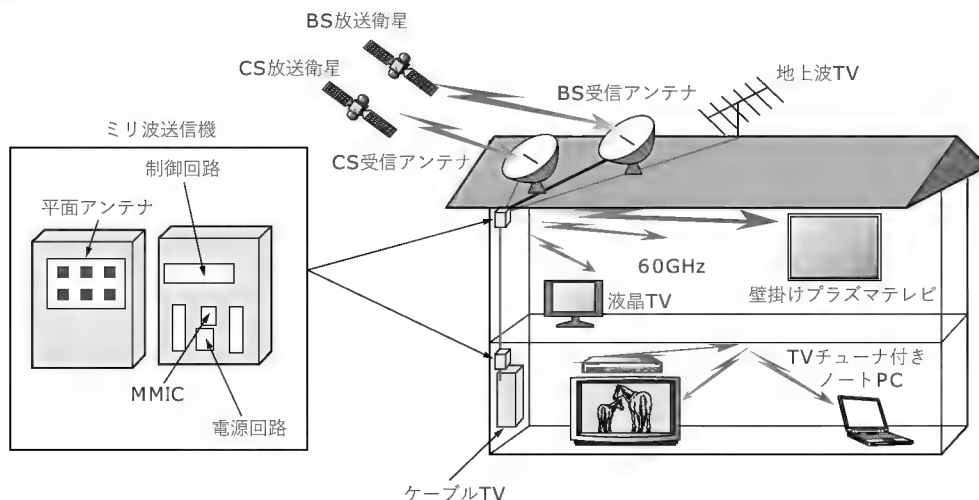
そこで，60GHz帯の無線伝送システムが世の中に普及するきっかけとなるような，周波数の広帯域性を最大限活用すると同時に普及が見込める低コストで簡易な60GHz帯の無線伝送システムが提案されています．これが屋内向け「ミリ波映像多重伝送システム」であり，通信総合研究所と民間企業数社が「YRPミリ波映像多重伝送システムに関する共同研究グループ」を設立して検討を行いました[注1]．

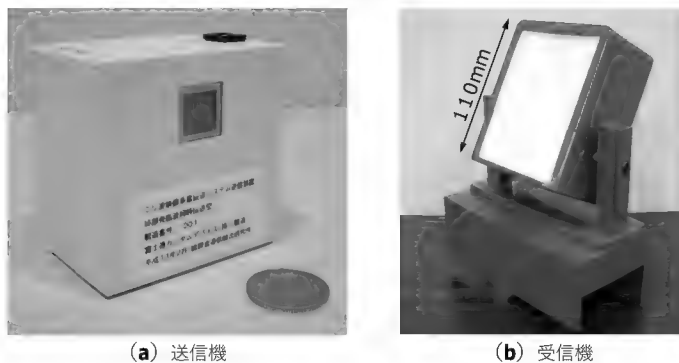
近年の放送サービスでは，サービスの多様化が進み，従来からの地上波放送サービスに加えて，BS放送やCS放送といった複数の衛星放送サービス，さらにケーブルテレビ放送といったように，複数の放送サービスを各家庭で利用する状況にあります．一方，テレビ受像機に関しては，プラズマテレビや液晶テレビの開発と普及が着実に進んでおり，壁掛けテレビとしての利用形態や，テレビを移動させて好きな場所で見るといった利用形態への需要が高まっています．しかしながら，各放送サービスを享受するためには，サービスごとに受信アンテナからのフィーダ線やケーブルをテレビチューナに接続する必要があり，このアンテナ線の存在が壁掛けテレビに関しては景観を損ない，またテレビを移動させる際にはその移動範囲を制約するといった問題があります．

そこで，このアンテナ線をまとめて60GHz帯の無線リンクで置き換えることを目的とするシステムがミリ波映像多重伝送システムです．なお，2.4GHz帯の免許不要バンドを使用してテレビ信号を無線伝送するシステムがいくつかすでに商品化されていますが，BS放送においてはディジタルハイビジョン放送サービスが始まると同時に，時代は“テレビは一家に一台の時代”から，“一人一台のパーソナルテレビの時代”へと遷移しつつあります．このような傾向のなかで，2.4GHz帯や5GHz帯を使用する無線伝送システムでは利用可能な周波数帯域の制約があるため，より広い周波数帯域を必要とするディジタルハイビジョン信号の伝送や，複数のユーザーに対して異なるチャネルの信号を同時に配信することは限界があります．このようなアプリケーションに対しては十分な周波数帯域を利用できる60GHz帯を使って，屋内に対してテレビ信号をそのまま無線で再放送する形態が，もっとも単純で有効かと思われます．

6.2 ミリ波映像多重伝送システムの構成

「ミリ波映像多重伝送システム」の概念を**図12**に示します．地上波やBS放送，CS放送などの放送信号が屋外の受信アンテナで受信された後，これらの信号がIF帯において周波数多重されて，小型のミリ波送信機に入力されています．ミリ波送信機は，先に述べたミリ波自己ヘテロダイン伝送方式にしたがって，同IF帯の多重信号を60GHz帯へ周波数変換してアンテナより室内へ送信します．一方，受信チューナ機能を内蔵した各テレビ受像機にはミリ波受信機が内蔵もしくは取り付けられており，送信機より送信された信号を受信して，やはり先に述べたミリ波

〔図12〕ミリ波映像多重伝送システムの概念

注1：独立行政法人通信総合研究所は民間企業8社（NTTアドバンステクノロジ，沖電気工業，シャープ，三洋電機，日本無線，日立国際電気，富士通カンタムデバイス，キヤノン）と共同で横須賀リサーチパーク（YRP）研究開発協議会「ミリ波映像多重伝送システム」に関する共同研究グループを設立し，平成11年度～平成12年度の2年間にわたり，同システムの実用化をめざして，方式面，装置面，伝搬面の検討を行った．

〔図13〕開発したミリ波映像多重伝送システムの外観

(a) 送信機

(b) 受信機

自己ヘテロダイン伝送方式にしたがって，もとのIF帯の映像多重信号へダウンコンバートされたのち，これがテレビチューナへと入力されています．

このようにすることで，屋内にむけてテレビ多重信号を再放送するような形式となるため，本システムでは異なる場所の複数のテレビ受像機で同時に異なるチャネルを受信することが可能です．なお，繰り返しになりますが60GHz帯の電波は家屋内における一般的な内壁材程度であれば透過するので，テレビ受信に必要なアンテナ配線がされていない部屋で，隣の部屋に設置されているミリ波送信機からのテレビ信号を受信してテレビを楽しむといった使い方も可能になります．とりわけ，BS放送やCS放送については各部屋にそのためのアンテナ配線がされていることは少なく，これを解決する手段としても利用できます．

ところで，現在のBS放送波を全チャネル伝送するためには約300MHzの帯域を要し，さらに近い将来にはチャネル数が増加されて500MHz程度まで拡張されることが予定されています．また，CS放送にいたっては垂直と水平の両偏波を使用して映像多重信号を放送しているため，じつに1～2GHz以上の帯域が伝送に必要となります．

これほどの広帯域信号を無線で伝送するには，現行の電波法では60GHz帯を用いるほかに選択肢はありません．また，開発費および製造コスト的な観点からは，本システムが無線LANなどと異なり単行通信であること，送受信機回路において変復調器機能を装備する必要のないシステムであること，およびすでに述べたように，ミリ波帯のシステムにおいてもっともコスト高の要因となる高安定発振器実現の問題がミリ波自己ヘテロダイン伝送方式で解決可能であることなどから，最初に普及する60GHz帯を用いる無線伝送システムとしてもっとも有望なシステムであると思われます．

6.3 ミリ波映像多重伝送システムの開発例と残された課題

図13は「YRPミリ波映像多重伝送システムに関する共同研究グループ」で平成12年度に開発したミリ波送信機およびミリ波受信機の外観写真です．また，同開発システムの仕様は**表4**に示しています．前述したようにミリ波帯の電波は非常に直進性が強く，見通し通信を前提とするため，このミリ波映像伝送システムにおいても，送信機と受信機間の見通しリンクが人物などの通過によって遮断されることは，映像伝送が一時的に中断されることを意味します．送信機が十分に部屋の高所に設置されていたとしても，受信機が設置されるテレビ受像機は通常見る人の目の高さ付近にあるため，比較的高い確率でこのような遮断は発生すると考えられます(**図14**)．

〔表4〕開発したミリ波映像多重伝送システムの仕様

項目	値
空中線電力	10mW
送信局部発振周波数	59.01GHz
RF変調信号周波数	60.01～60.51GHz
送信空中線利得(送信アンテナゲイン)	5dBi
受信空中線利得(受信アンテナゲイン)	30dBi
入出力IF周波数レンジ	1.0～1.5GHz(BS-IF対応)
伝送可能距離	見通し20m以上

〔図14〕人物によるミリ波リンクの遮断

これは，ビデオ録画などをすることを想定するとやはり不都合な点でしょう．このような問題を残すことは，場合によっては一般ユーザーにミリ波帯の無線通信は使いものにならないといったイメージをもたせてしまう可能性があります．じつはこの問題は，前述のミリ波自己ヘテロダイン伝送方式とアンテナダイバーシティ受信方式を併用すれば，非常に簡易に解決されます．これについては後であらためて解説することにします．

なお，上記のとおり設立された「YRPミリ波映像多重伝送システムに関する共同研究グループ」では，方式面，装置面，伝搬面の各方面からの検討を2年にわたって行いましたが，その結果を反映させて，電波産業界(ARIB)標準として標準化するにいたっています[13]．

6.4 ミリ波自己ヘテロダイン伝送方式とアンテナダイバーシティ受信

先にあげたミリ波映像多重伝送システムなどのシステムにお

いて，人物などの遮蔽によって信号伝送が中断もしくは瞬断する問題を解決するために筆者らは，提案しているミリ波自己ヘテロダイン伝送方式の採用によって可能となる，非常に簡易なアンテナダイバーシティ受信方式を提案・検討しています[14]．これを適用することで，人物などの受信機遮蔽による信号伝送の中断と遮断がほとんど発生しないシステムを実現することが可能になります．

これを簡単に説明すると，2台のミリ波受信機をテレビ受像機に少し離して設置もしくは内蔵しておくことで，同時に二つの見通しリンクがさえぎられる確率を十分に小さくする原理によるものです(**図15**)．こういった方法は「アンテナ空間ダイバーシティ受信」と呼ばれ，その考え方自身はけっして新しいものではありません．以降では，その方法とこれをミリ波帯無線伝送システムへ適用する場合の問題点などについて解説します．

● アンテナダイバーシティ方式

アンテナダイバーシティ受信方式にはいくつかの方式がありますが，なかでも，切り替えアンテナダイバーシティ方式と合成アンテナダイバーシティ方式は，よく利用される方式です．両方式における簡単なシステム構成を**図16**および**図17**に示します．

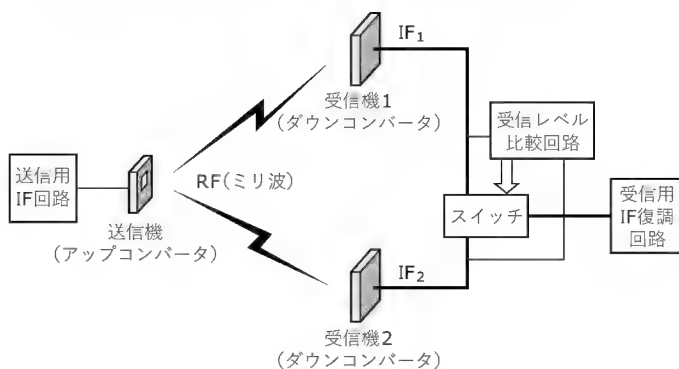
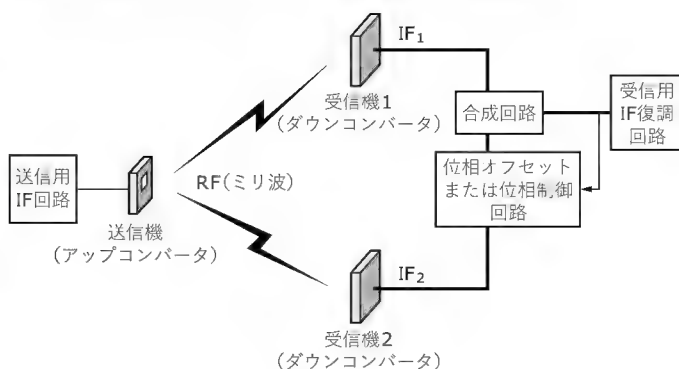
切り替えアンテナダイバーシティ方式では，受信側は受信機2台と，1台のスイッチ回路，およびこれを制御する信号レベル比較回路によって構成されます．通常はどちらかの受信機のみを使用して信号伝送を行い，何らかの理由で使用中のリンクが切れた場合にはそれを検知して，使用する受信機を他方のものへスイッチで切り替えます．

これに対し，合成アンテナダイバーシティ方式では受信機2台と1台の合成回路，および位相制御回路によって構成されます．ここでは常に2台の受信機が使用されており，その各受信機出力を合成したものを受信信号として復調します．合成アンテナダイバーシティ方式においては，信号を合成する際に位相が逆相で合成されると信号が打ち消しあってしまうため，同相合成を行うための位相制御が必要となります．この制御が比較的高度な技術となることから，合成アンテナダイバーシティ方式のほうが切り替えダイバーシティ方式より実現が難しく，コスト高になるとされています．しかしながら信号品質という観点からは，切り替えダイバーシティは切り替えによるスイッチングノイズや，非常にわずかな時間ながら信号伝送の瞬断が発生するため，合成アンテナダイバーシティ方式が望ましい方式といえます．

● ミリ波帯でのアンテナダイバーシティ実現の問題

アンテナダイバーシティ方式をミリ波無線伝送システムに採用する場合，ダイバーシティ効果を出すために，ある程度受信機間の距離を離す必要があります．このとき，ミリ波帯の高い周波数のままで受信信号を引きまわしたり，切り換えや合成などの操作を行うことは，伝搬ロスが大きくなることと回路に非常に微細な精度が必要とされることからあまり好ましくありません．したがって，一度ミリ帯の信号をIF帯に周波数変換した信号に対し，これらの処理を行うことが現実的です．しかしながら，先に解説したスーパーヘテロダイン伝送方式を用いた場合，各ミリ波受信機出力のIF信号には独立に大きな周波数ドリフトや位相雑音が発生しているため，これを同相合成することは非常に困難といえます．また，たとえスイッチングノイズや信号伝送の瞬断が発生することを妥協して切り替えダイバーシ

〔図15〕アンテナダイバーシティによるミリ波リンク遮断の防止

〔図16〕切り替えアンテナダイバーシティ受信システムの構成

〔図17〕合成アンテナダイバーシティ受信システムの構成

〔図18〕ダイバーシティ受信を用いた実証実験のようす

〔図19〕縦系テレビ信号配信システムの概念

ティ方式を採用したとしても，切り替えた後に非常に大きな周波数オフセットが生じている可能性が高く，これに対して瞬時に復調回路が応答できない可能性が高くなります．この結果，かなりの長時間映像が中断する結果となります．

これらの問題は，ミリ波自己ヘテロダイン伝送方式を採用することで解決されます．この伝送方式では，複数の受信機を独立に設置した場合でも，各受信機から得られるIF出力信号は単一の送信機へ入力されたIF信号のものからまったく周波数オフセットが生じていないものが得られるため，これらを同相合成することは容易になります．これを証明するため，テレビ受像機側で二つのミリ波受信機を1m程度離して設置し，各受信機より得られるIF出力信号を長さの等しいケーブルで電力合成器に入力する，たいへんシンプルなシステムを構成しました．このような構成でも，たとえ人物などが受信機を横切ったとしても画像伝送が瞬断することのない，たいへんスムーズなダイバーシティ受信が可能であることが確認されました．このとき使用した受信側のようすを**図18**に示します．

なお，複数の受信機を用いることは当然ながらシステムコスト高の要因となり得ますが，ミリ波自己ヘテロダイン伝送方式では受信機においてローカル発振器を内蔵する必要がないため，とりわけ受信機の低コスト化が可能な方式です．この点においても，ミリ波自己ヘテロダイン伝送方式は有利であるといえます．

7 縦系テレビ信号配信システム

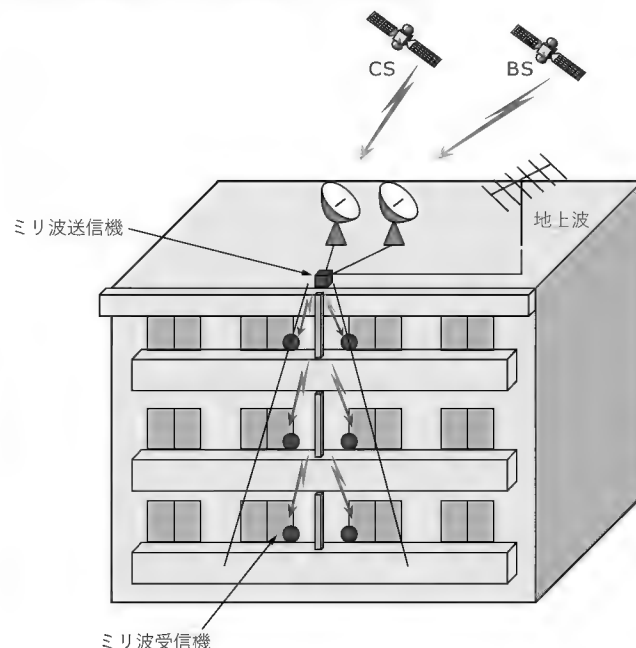
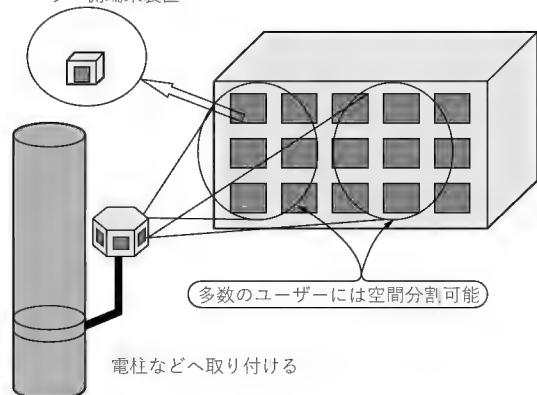
先に解説した屋内向けのミリ波映像多重伝送システムを屋外仕様に変更して，これをマンションなど共同住宅における共聴システムや広帯域信号配信のためのインフラ設備として利用する検討が現在進んでいます．このようなシステムを「縦系テレビ信号配信システム」と呼び，現在，独立行政法人通信総合研究所が「集合住宅へのミリ波帯電波を利用した縦系配線システムに関する調査検討会」を発足させておもに検討を行っています[注2]．

近年，一部の集合住宅では，共同受信設備が衛星放送に対応していない，南西に面したベランダがなく受信アンテナの設置が困難，衛星方向に電波をさえぎる建物が存在するなどの理由から，衛星放送サービスを享受できないという問題があります．そこで本システムでは，屋上に共同視聴用の衛星放送受信アンテナとミリ波送信機を設置し，各世帯にはミリ波受信機をベランダなどに設置することで各世帯への放送信号配信を可能にすることを目的としています．

図19にそのシステム概念を示します．本システムが屋外仕様であることを除けば，システムの構成や原理については先に紹介したミリ波映像多重伝送システムとまったく同等のものを適用することが可能です．しかし，本システムで用いる送信アンテナについては放射パターンが比較的狭いものでも十分と思われるため，送信側についても高いアンテナゲインをもつアンテナが使用できます．このことから，10階建て以上の高層マンションでも充分適用が可能と予想されます．なお，前述した調査検討会では本システムの実現に向けて，技術面，アプリケーション面，設置方法に関する課題の検討や意見交換を行い，およそ1年間の活動を計画しています．またその成果は規格化などへ反映し，製品実用化，市場への普及をめざすことになっています．

注2：独立行政法人通信総合研究所は11機関（埼玉大学，日本放送協会，JSAT，富士通カンタムデバイス，NECラボラトリーズ，沖電気工業，村田製作所，日立国際電気，マスプロ電工，京浜急行電鉄，総務省）とともに，「集合住宅へのミリ波帯電波を利用した縦系配線システムに関する調査検討会」（座長：埼玉大学　小林禧夫教授）を発足させた（平成14年7月）．

〔図20〕マルチポイントアクセスシステムのイメージ

8 100Base-TX対応高速無線リンクの開発

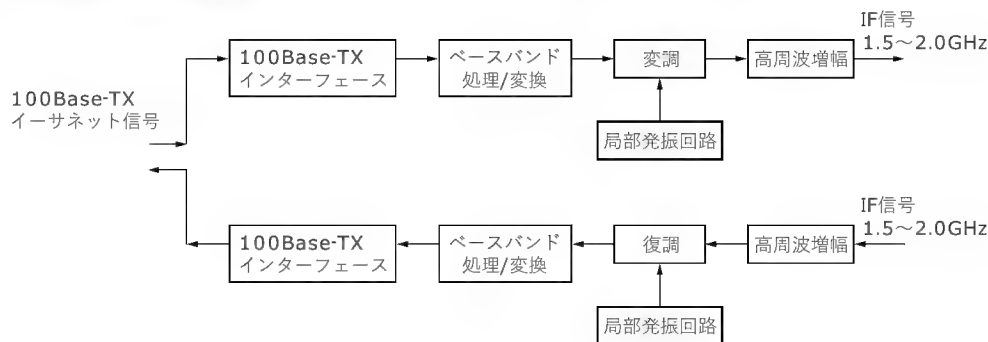
ミリ波自己ヘテロダイン伝送方式を利用して開発した，低コストな**100Base-TX**に接続可能な無線リンクシステムを紹介します．

通常，100Mbpsクラスの高速な伝送を実現するためには，高度な変復調技術や周波数の安定化技術が必要となることは前述のとおりです．しかし，開発したシステムはミリ波帯の広帯域の周波数が利用可能であることと，波長が短いため狭いアンテナビームを小型かつ容易に実現でき，**マルチパス**の影響を受けにくくできるという利点を生かし，単純なディジタル変調方式と100Base-TXのプロトコルをそのまま利用した単純な構成で実現できました．

このシステムは**図2**の中の一例にも示されているビル間通信のようなP-P（ポイントツーポイント）方式の無線通信リンクや，加入者系無線アクセスシステム（FWA：Fixed Wireless Access）などにおける無線LAN間のリンクとして利用できます．さらに，**図20**に示すようにP-M（ポイントツーマルチポイント）としての発展も可能です．

8.1 システム構成

今回開発したシステムでは，LANなどの通信インターフェースとして現在標準的に使用されている100Base-TXから受け渡されるイーサネット信号を2値位相変調（BPSK）に変換し，これを自己ヘテロダイン伝送方式を採用した低コストミリ波送受信モジュールを搭載した無線端末間で伝送させることで通信を行っています．このような方法で，100Base-TXインターフェースに対応した60GHz帯ワイヤレスリンクを低コストに実現しています．開発した100Base-TX対応ワイヤレスリンクシステムの全体構成を**図21**に，本構成図におけるベースバンド処理/IF変復調回路の構成を**図22**に示します．また，端末の外観を**図23**に示します．なお，ミリ波帯のRF回路となるミリ波送受信モジ

〔図21〕100Base-TX対応ワイヤレスリンクシステムの構成

〔図22〕ベースバンド処理/IF変復調回路の構成

〔表5〕開発した無線リンクのおもな仕様

局部発振周波数	59.01[GHz]
IF中心周波数	1.5[GHz]
送信用アンテナ利得	5[dBi]
受信用アンテナ利得	30[dBi]
変調方式	BPSK
インターフェース	100Base-TX

〔表6〕フレームサイズ対パケットロスの測定結果(伝送距離15m)

フレームサイズ[バイト]	64	128	256	512	1024	1280	1518
パケットロス[%]	0.000	0.000	0.000	0.026	0.009	0.000	0.000

ュールは，先に示したミリ波自己ヘテロダイン伝送方式の実証実験で使用したものと同等のものを使用しています(**図4**)．開発した装置のおもな仕様を**表5**に示します．**図23**に示したシステムでは，高利得受信アンテナとして導波管アレーアンテナを使用し，なおかつ防滴のための筐体を用意したために形状が少し大きくなってしまいましたが，アンテナの種類を変更するなどにより，さらなる小型化が可能と考えられます．

伝送性能の検証結果の一例として，端末間の距離を15mとし，伝送パケットロスを測定した結果を**表6**に示します．これより，どのフレームサイズでもほぼパケットロスがないことが確認されました．

〔図23〕100Base-TX対応ワイヤレスリンクシステムの外観

おわりに

本章では，最近注目されているミリ波帯の60GHzをとりあげ，60GHz帯の電波を使用する無線伝送技術について，電波法からの技術基準，標準化動向，デバイス製作とシステム開発における課題，課題を克服するためのミリ波自己ヘテロダイン伝送方式について解説し，同伝送方式を適用した無線アプリケーションとして検討されている，屋内向けミリ波映像多重伝送システム，縦系テレビ信号配信システム，100Base-TX対応高速ワイヤレスリンクの開発状況について解説しました．

近年の急速な無線システムの需要にともなって，現在普及している2.4GHz帯や5GHz帯などの免許不要バンドを使用する無線システムでは，近い将来，深刻な周波数不足と干渉問題が生じると思われます．そこで，次なる利用可能な免許不要バンドとして，60GHz帯の電波を用いる無線伝送システムが今後急速に普及すると期待されます．

参考文献

1) 電気通信技術審議会,『60GHz帯の周波数の電波を使用する無線設備の技術的条件』(平成12年2月28日答申)
2) Code of Federal Regulations, Title 47, Part 15, section 15.255, File: "47cfr15.pdf" at http://www.fcc.gov/Bureaus/Engineering_Technology/documents/cfr/1998/
3) File "fcc99183.pdf" at http://www.fcc.gov/Bureaus/Engineering_Technology/Notices/1999/
4) File "fcc00442.doc" at http://www.fcc.gov/Bureaus/Engineering_Technology/Orders/2000/
5) File "43116.pdf" at http://www.fcc.gov/Bureaus/Engineering_Technology/Documents/fedreg/61/
6) http://www.terabeam.com/products/equip/mmw.shtml
7) http://www.nokia.com/pc_files/MetroHopper.pdf
8) 平地ほか, "Status of millimeter-wave MMIC's and their applications in Japan", *Proceeding of GAAS 2000 - Paris 2000*, pp.369-372, Oct. 2000
9) 荘司ほか, "Millimeter-wave Remote Self-Heterodyne System for Extremely-Stable and Low-Cost Broad-Band Signal Transmission", *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES*, Vol.50, No.6, pp.1458-1468, (June 2002)
10) 荘司ほか, "60GHz band 64QAM/OFDM Terrestrial Digital Broadcasting Signal Transmission by Using Millimeter-wave Self-Heterodyne System", *IEEE TRANSACTIONS ON BROADCASTING*, Vol.47, No.3, pp.218-227(Sept. 2001)
11) 荘司ほか, "送信電力配分問題を最適化したミリ波自己ヘテロダイン方式による60GHz帯屋内BSディジタル放送伝送システム",『映像情報メディア学会論文誌』, 2002年8月号, Vol.56, No.8, pp.1334-1341 (Aug. 2002)
12) 報道発表, "ミリ波自己ヘテロダイン通信システムの開発に成功——高効率ディジタル変調信号の伝送が可能なミリ波通信システムの低コスト化に向けて——", http://www2.crl.go.jp/pub/whatsnew/press/ 000912/000912-1.html(Sept. 2000)
13)「特定小電力無線局ミリ波画像伝送用無線設備」ARIB STD-T69, 平成12年, (社)電波産業会
14) 荘司ほか, "Millimeter-wave Self-heterodyne Transmission Technique and a Simple Millimeter-wave Diversity Reception System", 2002 IEEE Radio and Wireless Conference(RAWCON), pp.115-118(Aug. 2002)
15) 報道発表, "ミリ波電波を利用した集合住宅内配線システム実現に向け，調査検討会が発足——衛星放送視聴困難なマンション住民に朗報——", http://www2.crl.go.jp/pub/whatsnew/press/020723-2/020723-2.html(July 2002)

しょうじ・ようぞう/つじ・ひろゆき　独立行政法人 通信総合研究所

数GHzの広帯域を使い，低消費電力で100Mbps以上の伝送速度を実現する

Chapter 5

超広帯域(UWB)ワイヤレス通信の基礎と動向

● 河野隆二

ワイドバンドCDMAによる第3世代移動通信システムIMT2000やIEEE802.11a/b/gなどのワイヤレスLANの研究開発と普及を通して，ユビキタスでよりブロードバンドなワイヤレス通信への関心が高まっている．その渦中で，2002年2月14日に米国FCC(連邦通信委員会)が数GHzの超広帯域，すなわちUltra Wideband(UWB)のワイヤレス技術を用いた製品の販売と利用を許可することを報道した．これを契機に，Bluetoothなどの近距離ワイヤレスシステムの市場で，はるかに高速(100Mbps以上)なシステムが実現できるUWB技術に，急速に注目が集まるようになった．

とくに，IEEE802.15.3aとして無線PAN(Personal Area Network)の高速版(WiMediaのAlternate)にUWBが採用され，IEEE802.15.xで規定される応用の中で最高速となり，その市場の広がりが期待されている．本章ではUWBとは何かという基礎から，今後の技術的課題と法制度化を含む今後の発展性について解説する．(筆者)

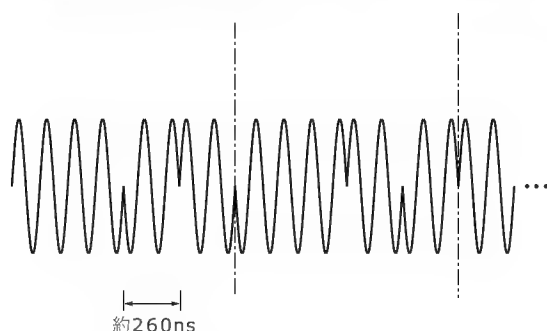
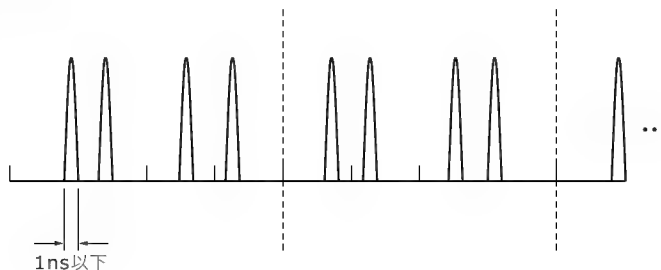
1 UWBとは何か？

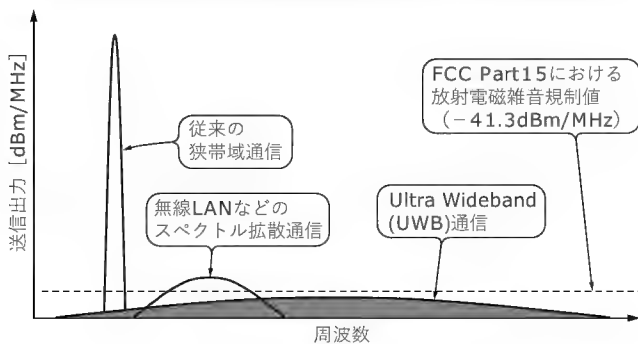
UWB(Ultra Wideband)ワイヤレス通信とは，搬送波(情報を変調するためのコサイン波)を用いず，1ns(10^{-9}秒)以下の非常に時間幅の短いパルスを用いて通信を行う方式であり，「Impulse Radio」とか「Time Domain」とか「Carrier Free」とも呼ばれる．そのため，帯域幅は数GHzにわたる超広帯域なものとなり，マルチパス歪みに強く，距離も測れるなどのスペクトル拡散通信方式と同様で，より顕著な利点がある．

図1に，インパルス無線(Impulse Radio)によるUWB信号波形を，通常の2相位相変調(BPSK：Binary Phase Shift Keying)信号波形と比較して示す．図に示すように，UWB方式ではコサイン波のような搬送波による変調をせずに1ns以下のパルスを複数送信する．そのため，占有帯域幅は非常に広くなり，スペクトル電力密度は非常に小さくなるため，通常のスペクトル拡散通信方式と同様に秘話性・秘匿性に優れ，他の狭帯域通信に与える影響は小さいなどの特徴をもつ通信方式である．超広帯域に拡散されるので，よりその特徴がさらに強調される．

図2に示すように，UWB信号はBPSKなどの被変調信号はもとより，通常のスペクトル拡散信号(2.4GHz帯ワイヤレスLANで数十MHzの帯域幅)に比べても超広帯域(数GHzの帯域幅)であることから「UWB」と呼ばれ，電力スペクトル密度もスペクトル拡散信号(2.4GHz帯のワイヤレスLANで10mW/MHz以下)に対し，UWB信号(1MHzあたり10nW：10nW/MHz以下)ははるかに低く，他のシステムが共存しても干渉を与えにくいばかりでなく，他のシステムからの干渉にも耐えられ，スペクトル拡散信号の特徴を強調した利点があるといえる．

さらに，パルスの時間幅が非常に小さいため，マルチパスを細かく分解でき，RAKE受信(複数方向から届く電波を集めて受信すること)が可能となるので，マルチパスにも強い方式である．また，UWBでは1ns以下のパルス(モノサイクル)からなるベースバンド信号を複数用意する．そのモノサイクルをユーザーごとに固有のタイムホッピング(TH)系列分だけシフトさせて送信する．そのため，多元接続する際，他局のモノサイクルが

〔図1〕Impulse Radioによる UWB信号と搬送波を用いた位相変調信号の比較

(a) 2相位相変調(BPSK)による時間信号波形

(b) Impulse RadioによるUWB時間信号波形

〔図2〕UWB信号と従来の変調信号の電力スペクトル分布の相違

衝突(ヒット)したときに誤りとなり，このヒットする確率が誤り率などのシステム性能を左右する要素となる．

2 UWB通信方式の送信から受信までの処理

2.1 UWBの送信信号

k番目のユーザーの送信信号のブロック図を**図3**に示す．k番目のユーザーの送信信号は，次のように表される．

$$s_{tr}^{(k)}\left(t^{(k)}\right)=\sum_{j=-\infty}^{\infty} w_{tr}\left(t^{(k)}-jT_f-c_j^{(k)}T_c-d_j^{(k)}\right) \cdots\cdots\cdots\cdots (1)$$

ただし，$t^{(k)}$は送信機のクロックタイム，T_fはパルス反復時間，T_cはTH(Time Hopping)のチップ長，$c_j^{(k)}$はk番目のユーザーのj番目のTH系列，$d_j^{(k)}$はk番目のユーザーのjホップ目の情報系列，$w_{tr}^{(t)}$は送信機のモノサイクル波形である．

k番目のユーザー送信機からはそれぞれ違う時間だけシフトされた複数のモノサイクルが送信される．たとえば，j番目のモノサイクルは$jT_f-c_j^{(k)}T_c+d_j^{(k)}$の時間に送信されはじめる．各シフト時間の構成は，次のように述べられる．

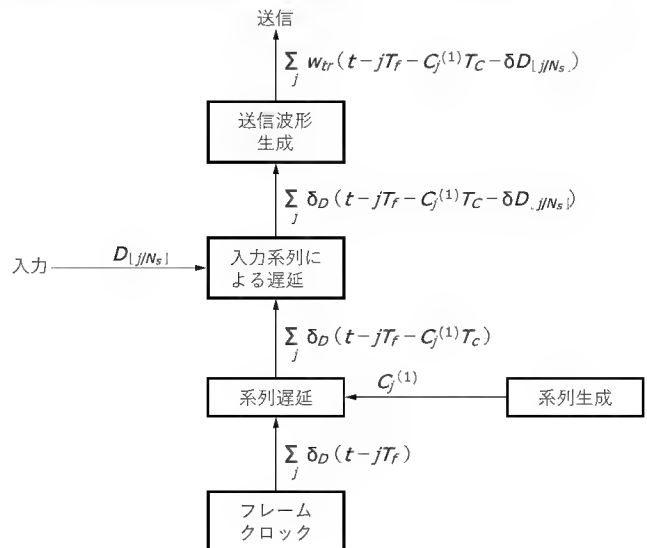
1）一定間隔のパルス列

$$\sum_{j=\infty}^{\infty} w_{tr}\left(t^{(k)}-jT_f\right)$$

で表されるパルス列は，T_f秒ずつ離れたモノサイクル列によって構成される．一定間隔のパルス列によって多元接続する場合，他局からの信号がすべてのパルスで衝突してしまうことが起こり得る．

2）ランダム/疑似ランダムTH

多元接続において，1)のようにすべてのパルスで衝突することがないように，ユーザーごとに異なるシフトパターンのTH系列$\left\{c_j^{(k)}\right\}$を用いる．このホッピング系列$\left\{c_j^{(k)}\right\}$は系列長N_pの疑似ランダム系列で，各要素は$0 \leqq c_j^{(k)} < N_h$を満たすものとし，$N_hT_c \leqq T_f$を満たすようにN_hを定める．このTH系列はパルス列の各パルスを付加的に時間シフトさせる．つまり，j番目のパルスは1)に加えて$c_j^{(k)}T_c$秒シフトされる．このシフト時間は0からN_hT_cまで離散的な値をとる．疑似ランダムTH系列の系列長はN_pなので，

〔図3〕UWB送信側のシステム構成

$$\sum_{j=-\infty}^{\infty} w_{tr}\left(t^{(k)}-jT_f-c_j^{(k)}T_c\right)$$

で表される波形は，$T_p=N_pT_f$の周期をもつ．

$N_hT_c \leqq T_f$となるので，N_hT_c/T_fはフレーム時間T_fの中でTHが許される割合を表す．モノサイクル相関器の出力を読み取るためにわずかに時間間隔が必要なため，N_hT_c/T_fは1より小さくなる．

N_hT_c/T_fが小さすぎると，大衝突が起きやすくなる．逆に，N_hT_c/T_fが十分大きく，TH系列がきちんと設定されていると，多元接続による他局間干渉はガウス分布に近似される．

3）情報系列

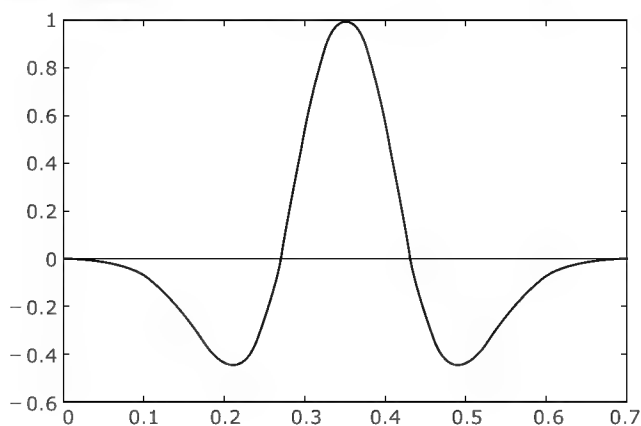
送信データを表す$d_j^{(k)}$は，$d_j^{(k)}=\delta D_{\lfloor j/N_s \rfloor}^{(k)}$と表される(ここで$\lfloor x \rfloor$は$x$の整数部分を表す)．ここで，$\left\{D_i^{(k)}\right\}_{i=-\infty}^{\infty}$で表される系列は0，1のバイナリ情報系列であり，$\delta$はパルスの時間幅$T_m$によって決まる値である．UWBは1シンボルで$N_s$個のモノサイクルを送信するので，データ系列は$N_s$のホッピングごとに変化する．データシンボルが$j=0$から始まるとすると，データシンボルの順番はホッピング回数jを使って$\lfloor j/N_s \rfloor$と表される．この変調方式では，データシンボルが0のときはモノサイクルにシフト時間を追加されず，データシンボルが1のときはモノサイクルにΔのシフト時間が追加される．

4）多元接続するとき

多元接続する際，N_uのユーザーが接続しているときの受信信号は，

$$r(t)=\sum_{k=1}^{N_u} A_k s_{rec}^{(k)}\left(t-\tau_k\right)+n(t) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

と表される．ここでA_kはk番目のユーザーからの受信信号$s_{rec}^{(k)}(t-\tau_k)$が伝搬路を通って減衰した値を示す．また，τ_kは受信器のクロックとk番目のユーザーのクロックの非同期の時間を表し，$n(t)$は他局間干渉以外の雑音を表す．

〔図4〕送信波形

理想的なチャネルとアンテナシステムでも，送信波形 W_{th} は受信器のアンテナの出力では W_{rec} に変化する．典型的な理想化されたモデルでは，受信波形 W_{rec} は**図4**のように表される．解析を容易にするために，受信波形 W_{rec} は既知のものとし，マッチドフィルタで受信することを考えた．

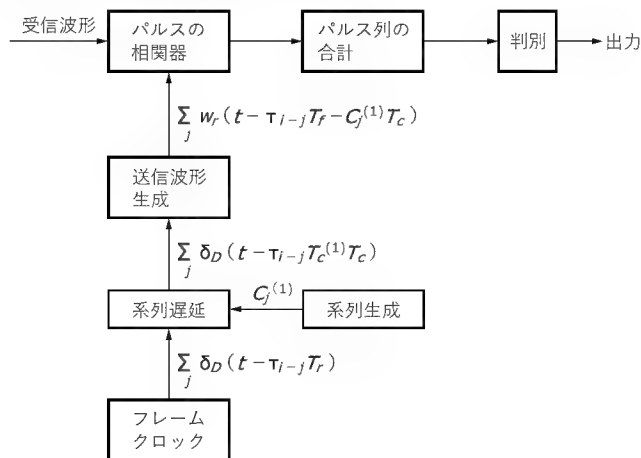
2.2 UWBの受信器

UWBの受信器において同期が完全であると仮定する．UWBの受信器のブロック図を**図5**に示す．UWBの受信器は，$T_s=N_sT_f$ の時間で受信信号 $r(t)$ を観測して $D_{(0)}^{(1)}$ が0か1かを決定する必要がある．つまり，送信情報が d のときの受信信号の状態を H_d と表すと，

$$H_d : r(t) = A_1 \sum_{j=0}^{N_s-1} w_{rec}\left(t-\tau_1-jT_f-c_j^{(1)}T_c-\delta d\right)+n_{tot}(t) \quad \cdots\cdots\cdots\cdots(3)$$

の d が0か1かを判別する必要がある．(3)式の $n_{tot}(t)$（他局間干渉と受信雑音）はまとめて，

$$n_{tot}(t) = \underbrace{\sum_{k=2}^{N_u} A_k s_{rec}^{(k)}(t-\tau_k)}_{\text{他局間干渉}} + \underbrace{n(t)}_{\text{受信雑音}} \quad \cdots\cdots\cdots\cdots(4)$$

〔図5〕UWBの受信側のシステム構成

と表される．

他ユーザーがなく，かつ，データ $D_{(0)}^{(1)}$ が独立な乱数の場合，最適受信機は相関受信機で，

$$D_0^{(1)} = 0 \Leftrightarrow \underbrace{\sum_{j=0}^{N_s-1} \overbrace{\int_{\tau_1+j\tau_f}^{\tau_1+(j+1)T_s} r(t)\, v\left(t-\tau_1-jT_f-c_j^{(1)}T_c\right)dt}^{\text{パルス相関器の出力}=\alpha j}}_{\text{相関器出力の合計}=\alpha} > 0 \quad \cdots\cdots\cdots\cdots(5)$$

と表される．ここで，

$$v(t) \triangleq w_{rec}(t) - w_{rec}(t-\delta) \cdots\cdots\cdots\cdots(6)$$

である．$w_{rec}(t)$ は $[0,\ T_m]$ の期間で0でないので，$v(t)$ は $[0,\ T_m+\delta]$ の期間で0でない．式(5)の α は受信信号 $r(t)$ に合わせてホッピングした相関波形 $v(t)$ を用いてとった各パルスの相関値の合計値である．**図3**を用いて得られる相関器の波形を**図6**に示す．

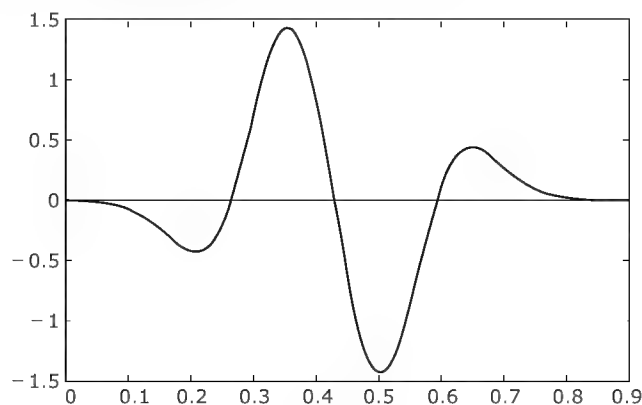
厳密にいえば，以上の方式は他ユーザーの信号がある場合は最適ではない．ガウス分布でない多元接続による雑音が存在する場合，最適受信機では多元接続雑音の構成の情報を利用する．したがって，マルチユーザー環境では最適受信機構成はさらに複雑になる．**図7**にUWBのマルチユーザー環境下における相関受信を示す．同時に多元接続するユーザー数が増え，マルチユーザー受信が不可能になってくると他局間干渉はガウス分布に近付いてくる．このような状況では，$n_{tot}(t)$ は白色ガウス雑音とみなされ，式(5)は最適となる．

α は $\alpha = m + n_d$ と置き直すことができ，希望信号の相関器出力 m（H_1 のとき）と干渉成分の相関器出力 n_d はそれぞれ，

$$m = \sum_{j=0}^{N_s-1} \int_{\tau_1+jT_f}^{\tau_1+(j+1)T_f} \left[A_1 \sum_{i=0}^{N_s-1} w_{rec}\left(t-\tau_1-iT_f-c_i^{(1)}T_c-\delta\right) \right] \times v\left(t-\tau_1-iT_f-c_i^{(1)}T_c\right)dt \quad \cdots(7)$$

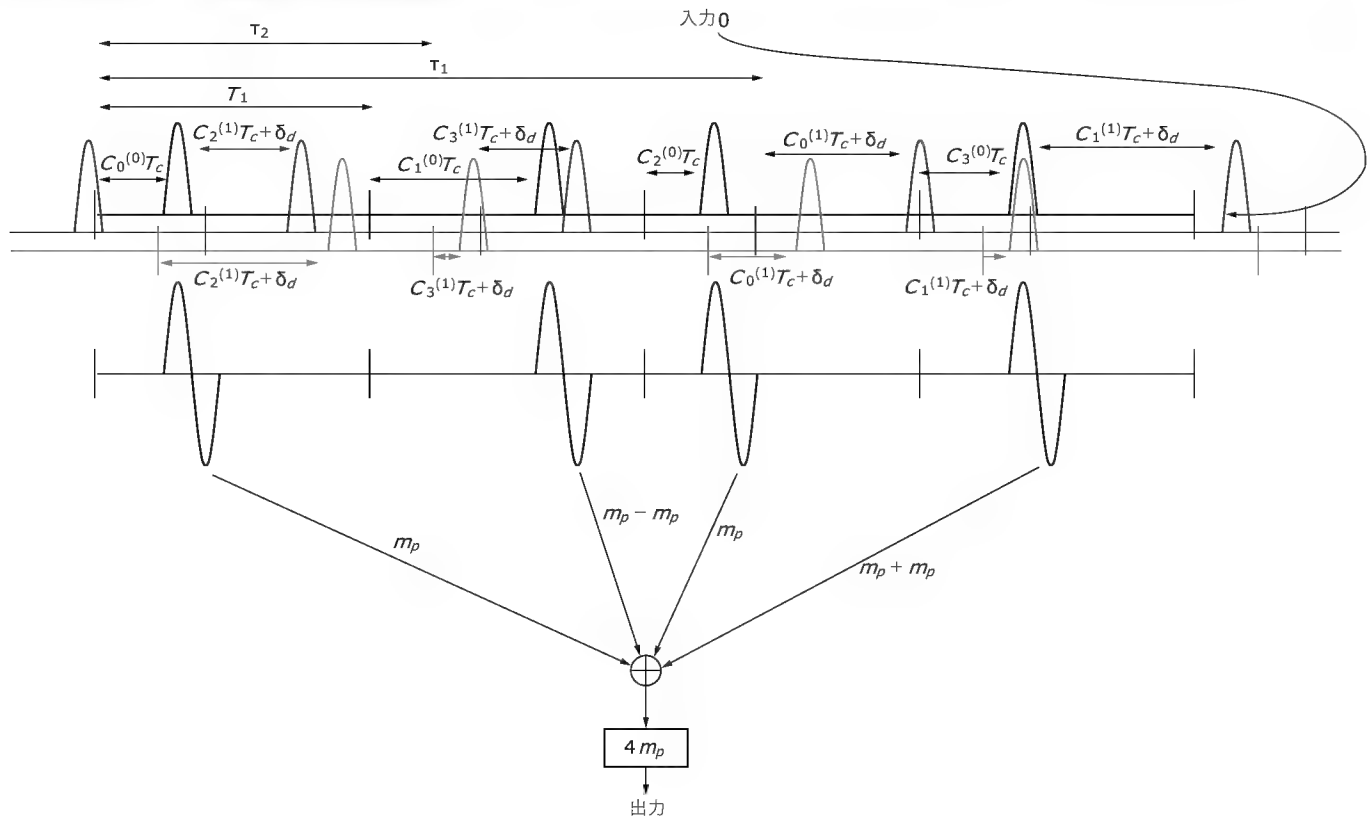
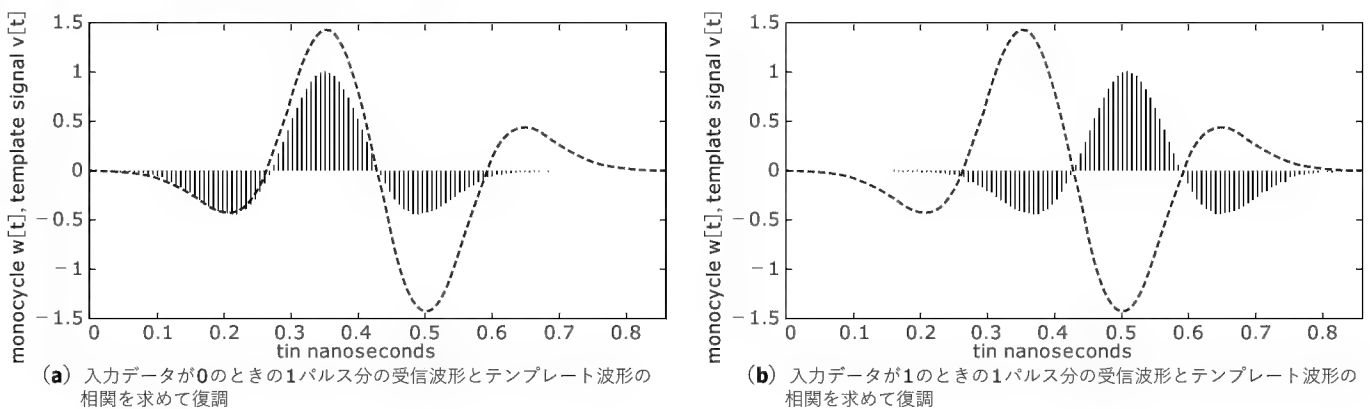
と，

$$n_d = \sum_{j=0}^{N_s-1} \int_{\tau_1+jT_f}^{\tau_1+(j+1)T_f} n_{tot}(t)\, v\left(t-\tau_1-jT_f-c_j^{(1)}T_c\right)dt \quad \cdots\cdots(8)$$

〔図6〕受信機で相関をとるテンプレート波形

〔図7〕マルチユーザー環境下のUWB相関受信

〔図8〕受信パルスとテンプレート波形の相関

(a) 入力データが0のときの1パルス分の受信波形とテンプレート波形の相関を求めて復調

(b) 入力データが1のときの1パルス分の受信波形とテンプレート波形の相関を求めて復調

である．mは$m=N_sA_1m_p$と表される．式(8)はさらに簡単に，

$$n_d=\sum_{k=2}^{N_u}A_kn^{(k)}+n_{rec} \quad \cdots\cdots(9)$$

と表され，$n^{(k)}$はk番目のユーザーからの他局間干渉を表し，n_{rec}はモノサイクル以外の原因による雑音を表す．

2.3 UWB受信機のSNR

UWBにおける信号対雑音電力比SNRは，

$$SNR_{out}(N_u)\triangleq\frac{m^2}{IE\{|n_d|^2\}} \quad \cdots\cdots(10)$$

と表される．$n^{(k)}$は平均0である．式(10)で示すn_dは独立な平均0の乱数なので，n_dの分散値$IE\{|n_d|^2\}$は，

$$IE\{|n_d|^2\}=\sigma_{rec}^2+N_s\sigma_a^2\sum_{k=2}^{N_u}A_k^2 \quad \cdots\cdots(11)$$

と表される．σ_{rec}^2は受信雑音の成分である．

希望信号だけが存在する場合，つまりN_u=1で多元接続しない場合のSNRは，

$$SNR_{out}(1)=\frac{(N_sA_1m_p)^2}{\sigma_{rec}^2} \quad \cdots\cdots(12)$$

と表される．また，N_uユーザーで多元接続する場合の$SNR_{out}(N_u)$は，

$$SNR_{out}(N_u)=\frac{(N_sA_1m_p)^2}{\sigma_{rec}^2+N_s\sigma_a^2\sum_{k=2}^{N_u}A_k^2} \quad \cdots\cdots(13)$$

と表される．

受信パルスとテンプレート波形の相関について，**図8**(前頁)に示す．

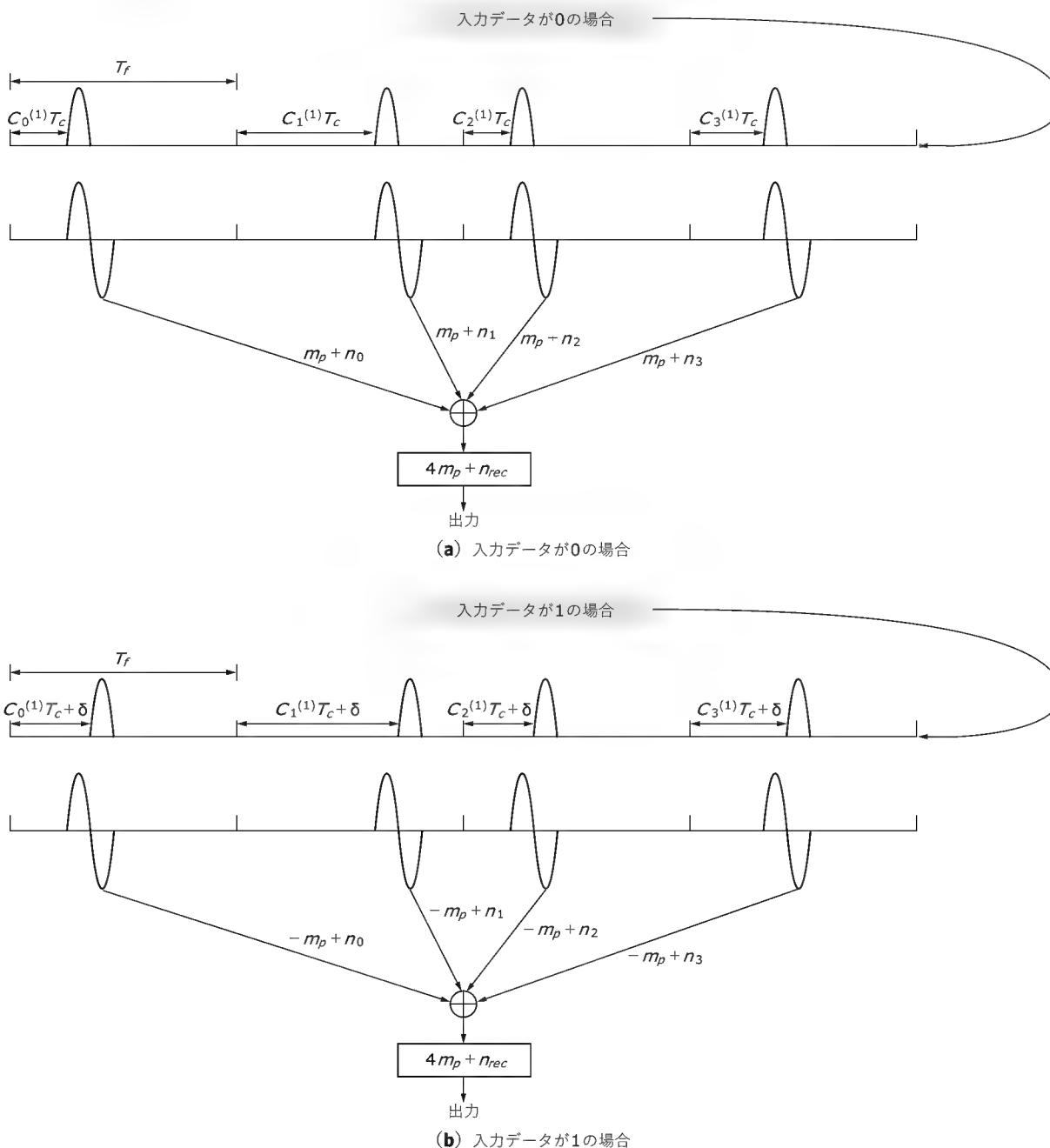
3 UWBの特徴

Impulse RadioによるUWB方式では，DS-SS方式のように拡散符号のチップの時間幅を小さくすることによって帯域を広げるのではなく，Time Hoping(TH)された1ns以下の時間幅をもったパルス(モノサイクル)のベースバンド信号を複数送信することによって，非常に広い帯域を占有するようになっている(**図9**)．したがって，搬送波を用いて変調するときよりもさらに広い帯域で，スペクトル電力密度を低減して伝送する．その結果，高い秘匿性や干渉への耐性，到達時間の分解能力を有する．

UWBでは，伝送する周波数帯に情報信号を乗せる搬送波を使わず，代わりにモノサイクルのベースバンド信号で伝送する点が特徴である．UWBでは占有帯域幅が数GHzにわたるため，良好な伝搬路特性でも歪みを生じてしまう．しかし，UWBでは

〔**図9**〕**Time Hopping**(TH)**/Pulse Positioning Modulation**(PPM)**によるUWBの送受信**
(ユーザーごとに固有のタイムホッピング(TH)系列分だけ遅延させたパルスを複数送受信する)

(**a**) 入力データが0の場合

(**b**) 入力データが1の場合

非常に広い帯域を用いるため，

① マルチパスの遅延時間を1ns以下に分解できる．この結果，マルチパスフェージングの影響を十分抑えることができる
② 高いパス分解能力により，UWBによる室内の高品質近距離無線通信が可能
③ フェージングの影響が低く抑えられ，送信電力が少なくてすむ
④ 送信電力の低いUWBでは電力スペクトル密度が非常に低くなるので，他の狭帯域伝送に影響をほとんど及ぼさない

といった特徴をもつ．

＜UWBの物理的特徴＞

① UWB信号
- パルス列であり，モノサイクルと呼ばれるパルス
- 余弦波による搬送波を用いない(広義で搬送波を使うものもあり)
- Impulse Radioとも呼ばれる
- 通常，パルス時間幅はpico secondsからnano seconds
- 典型的なパルス波形はGaussian
- パルス繰り返し周期は，通常，0.1秒間隔

② UWB帯域幅
- 比帯域幅BW＝(帯域幅)/(中心周波数)が，通常25％以上，

$$BW = 2\frac{f_n - f_L}{f_H + f_L} = \frac{f_n - f_L}{f_c}$$

- f_c = 2.4GHz，f_H = 3.0GHz，f_L = 1.8，1.2GHz(比帯域幅＝50%)

(例) 従来通信方式との比較
- AM　6.8kHz/530kHz = 1.3%
- cdmaOne　1.25MHz/800MHz = 0.15%
- W-CDMA 5MHz/2200MHz = 0.23%
- 無線LAN(IEEE802.11) 22MHz/2450MHz = 0.9%

③ 処理利得(Processing gain ： PG)
- UWBシステムは，DS-CDMAシステムと，同一占有帯域幅で同程度の処理利得をもつ．したがって，干渉に対して強い

④ 通信容量

GHzオーダの帯域幅B(後述)を用いるUWBは，超高速伝送が高信頼度で可能．

＜UWBの実用上の特徴＞

① 電力スペクトル密度がきわめて低い(雑音レベル，DS-SS以下)
⇒既存の通信システムとの与干渉・被干渉が少なく，共存の可能性
② 平均電力レベルが1mW以下で数マイル伝送可能
③ きわめて短い(ns単位)のパルスを利用
⇒RAKE受信によりマルチパスに強い(高いパス分離能力)
⇒レーダとして高精度測距(数cm単位)が可能(高い距離分解能)
④ キャリアなし，信号の放射時間がきわめて短い
⇒小型・低消費電力のシステムを構築可能
⑤ 常に広い帯域を占有(GHzオーダ)
⇒大容量多元接続・超高速伝送(＜数百Mbps)可能
⑥ 通信と測距が同時にできる
⇒ITS(車車間通信など)に応用可能

＊　　　＊

以上のような特徴は，おもにスペクトル拡散通信方式の特徴と共通するものである．その理由は，信号の周波数スペクトル成分が広帯域に拡散されることにより得られる特徴として，UWBもスペクトル拡散方式と共通だからである．

スペクトル拡散の方法では，広帯域の拡散符号を情報信号にかけ算することで情報信号のスペクトルを拡散する直接拡散変調(Direct Sequence or Direct Spread ： DS)や，周波数シンセサイザを拡散符号で発信周波数(搬送周波数)を切り替えて(ホップさせて)情報信号のスペクトルを拡散する周波数ホッピング変調(Frequency Hopping ： FH)がおもに用いられる．

一方，Impulse RadioによるUWBでは，情報信号を超広帯域のスペクトルをもつパルス，またはパルス列に変換することにより，信号のスペクトルを広帯域に拡散している．

このように広い周波数帯域を用いると高速伝送が可能になる理由は，シャノンが導いた通信路容量Cによって説明できる．有線/無線のいずれの場合も，物理的に提供された通信路ごとに固有に誤りなく伝送できる最大の伝送速度として，通信路容量Cが定義される．とくに，伝送できる周波数帯域Bが制限され，雑音により誤りが生じる通信路では，通信路容量Cは，下記のように表される．

$$C = B\log_2\left(1 + \frac{P}{N}\right)$$

C=Maxchannel capacity in bits/s
B=Channel bandwidth in Hertz
P=Signal power in Watts
N=Noise power in Watts

つまり，最大伝送速度Cは通信路の帯域幅Bに比例して大きくでき，信号電力対雑音電力比(P/N)を大きくすれば，Cは大きくできる．したがって，UWBのようにGHzオーダまでBを広帯域にすることにとり，超高速伝送が可能となる．

一方，通信路でのシンボル誤りは，通常の無線通信では，雑音だけではなく，**図10**に示すように，壁などの障害物による電波の反射や回折などによる多重波伝搬，いわゆるマルチパス(Multi-path)による情報シンボル間干渉，さらに，**図11**(p.107)に示すように，複数のユーザーが無線通信路にアクセルする場合，いわゆる多元接続(Multiple Access)する場合にユーザー間のパルスが時間的に衝突するユーザー間干渉によって誤りが生じる．

UWBでは，おもに**図11(a)**に示すように，ユーザーごとにデータ1シンボルを伝送するパルス列の時間的位置を変えて，できるだけパルスの時間的衝突(ヒット)を避けるように時間ホッピングパターン(拡散符号に相当し，TH系列という)を各ユー

〔**図10**〕**ワイヤレス通信における多重波伝播**(マルチパス)

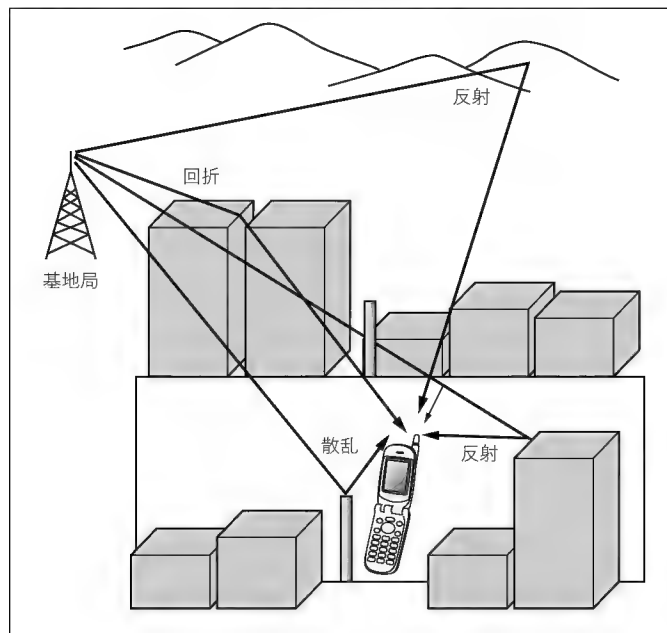
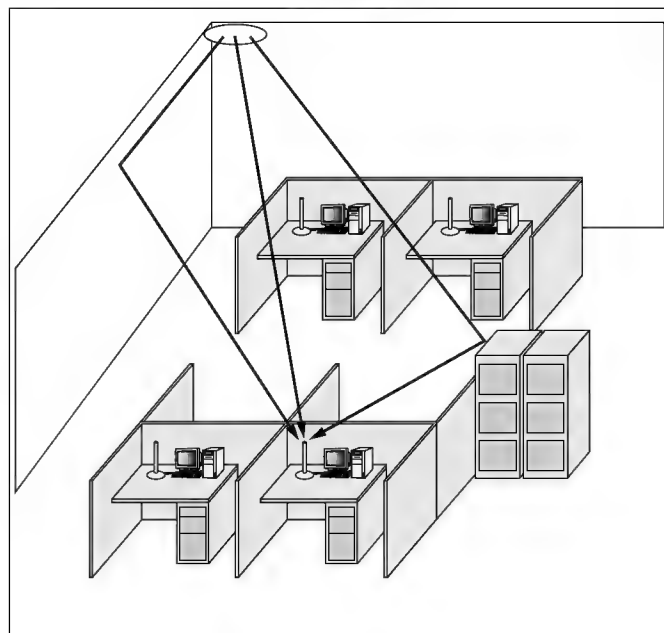
(屋外の伝搬路環境)

(屋内伝搬環境)

ザーに割り当て,CDMA(Code Division Multiple Access:符号分割多元接続)を行う.

図11(**b**)のように,ユーザーごとに異なるTH系列でタイムホップされたパルスを送受信するので,他局のパルスと希望局のパルスが衝突(ヒット)する確率がシステム性能を決める.しかし,伝送速度一定条件の下では,パルス間隔を広げることはできず,ヒット確率を抑えることはできない.伝送速度一定条件の下で,パルス間隔を広げることができれば,UWBのシステム性能は良くなる.

UWB技術は,歴史的には通信よりもパルスレーダとして研究開発されてきた.UWB信号で距離を測る原理はいたって簡単で,送信したパルス信号が障害物に当たり,その反射信号を受信するまでの時間を計ることにより,そのパルスの往復時間に電磁波の伝搬速度を掛ければ,往復距離が計算できる.とくに,遠くまで測距するためには,パルスの先頭値を大きくし,隣接する異なる物体を区別して認識できる距離分解能を精細にするためには,パルス幅を短くし,異なる物体からの反射パルスが受信端で区別して認識できればよい.

〔**図12**〕**送信パルス**(パルス信号の到着時間を計測することで,距離を測ることができる)

図12のように,UWBパルスは超短時間幅であるから,距離分解能に優れている.また,通信しながら,ユーザーごとに割り当てられたTH系列を受信端で相関を求めることにより,ユーザーごとの反射パルスを区別するだけでなく,1パルスの先頭電力が小さくても相関により複数パルスの電力を集約できるので,測距範囲を延ばせる.

以上のような特徴をUWBはもっているため,学術的にもスペクトル拡散技術との関係や性能解析,各種の改良方式などが必要である.応用面ではUWBの特徴を活かしたシステム開発が期待されている.

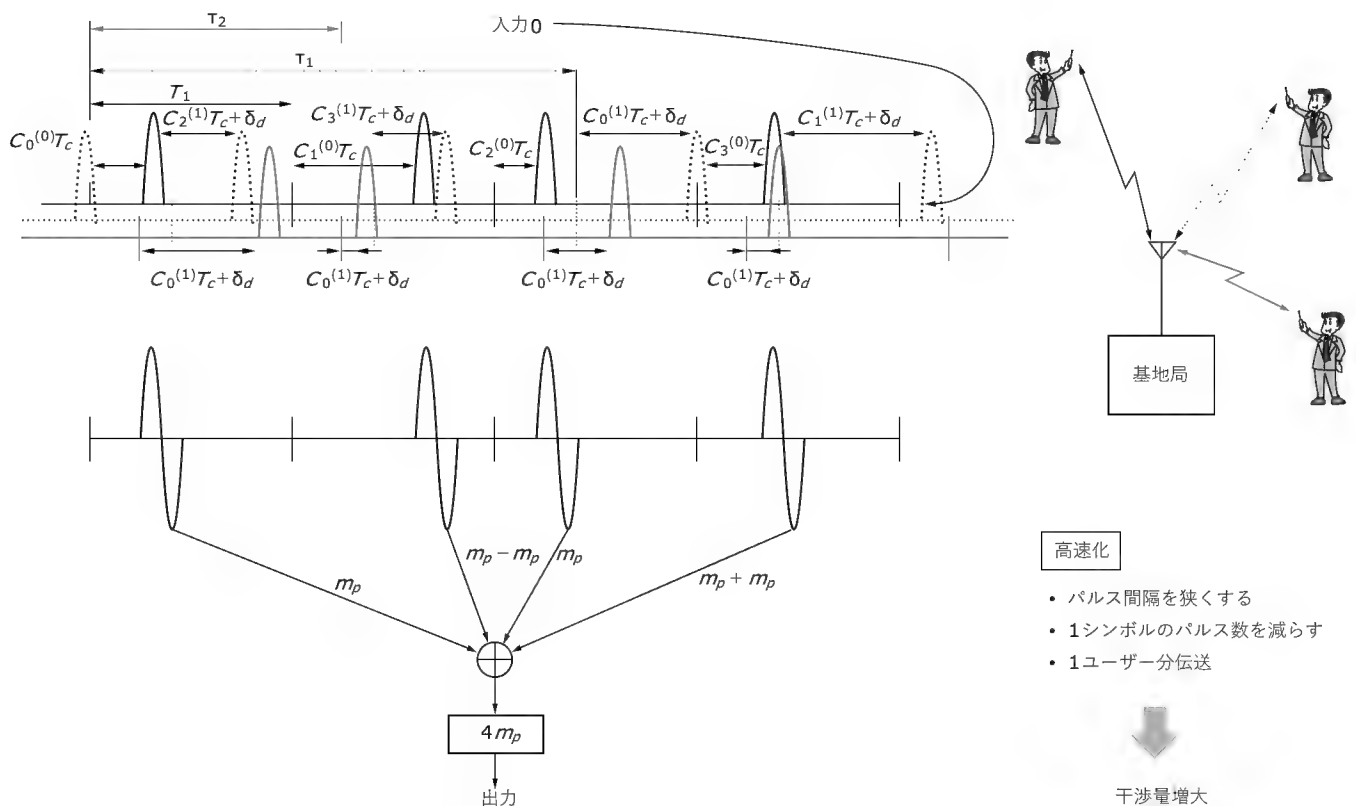
一方,UWBは従来,運用に研究開発されてきたが,商用化するためには検討すべき課題がまだ多い.UWBの問題点を下記に示す.

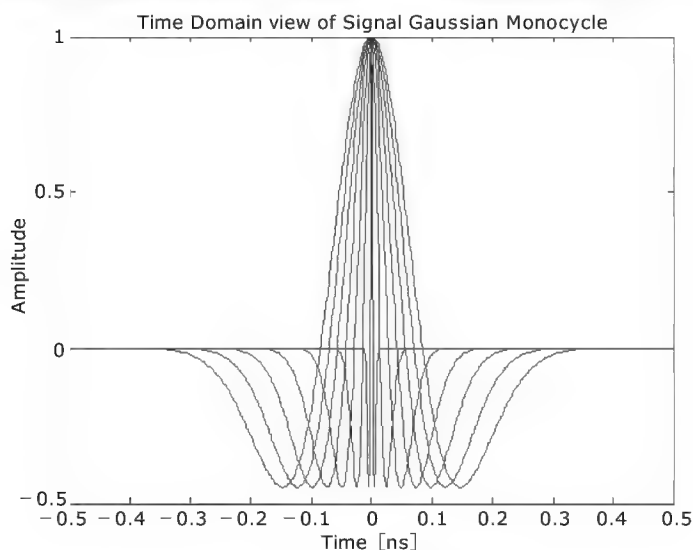
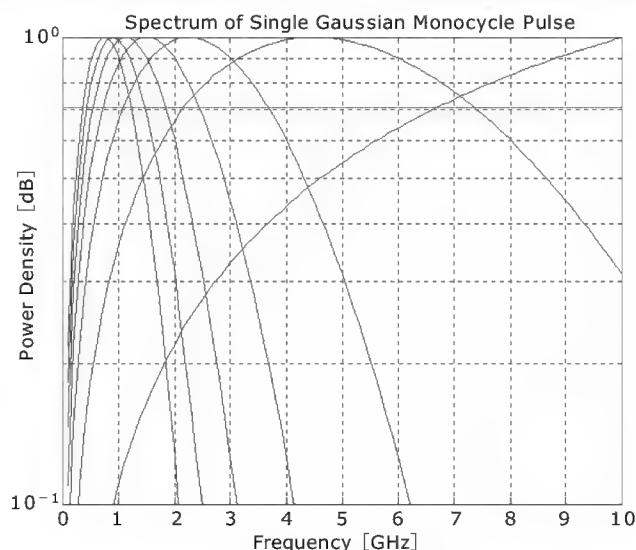
<UWBの問題点>

① 超広帯域で時間幅の非常に短いパルスを発生させる回路,素子,および超広帯域アンテナ・高周波回路の製造
② 受信時のパルス位置ずれの検出精度
③ マルチパス環境下でのパルス符号間干渉
④ マルチユーザー環境下でのパルス衝突によるユーザー間干渉(システム内干渉)
⑤ 周波数共用(共存システム)によるシステム間干渉
⑥ 超広帯域スペクトルを利用できる周波数帯が電波法上困難で

〔図11〕多元接続する場合

(各パルスをユーザーごとに違うタイムホッピング系列分シフトさせて送受信する．ただし，干渉ユーザーが増えるとパルスのヒット確率が増加し，相関器出力に干渉成分が多く入ることになる)

(**a**) UWBによるCDMA(ユーザーの分離)の原理

(**b**) UWB-CDMAにおけるユーザー間干渉

〔図13〕受信パルス(基本パルス波形によって，周波数分布が決定される．したがってパルス波形を変更することにより，周波数特性を制御できる)

(a) 受信パルスの時間波形

(b) 受信パルスの周波数スペクトル

あった

先に述べたように，UWBはスペクトル拡散通信方式の一種とみなすこともでき，スペクトルの拡散方法が直接拡散方式や周波数ホッピング方式と異なり，超短時間長のインパルスを用いる．そのハードウェア実現形式によって異なるが，従来のスペクトル拡散方式では，GHzオーダの超広帯域にスペクトル拡散することは困難である．すなわち，直接拡散方式でこのような超広帯域の拡散符号を生成する回路を構成するには，高価なハードウェアが必要である．また，周波数ホッピング方式でもそのような超広帯域に周波数をホッピングするシンセサイザは非常に高価なものとなる．したがって，UWBの特徴を活かすためには，超短時間長のパルスを生成するハードウェアとその受信検出器の構成の実現性にかかっているといえる．従来，軍用に研究開発されてきたが，商用化するためには検討すべき課題が未だ多い．

そのほか，インパルス無線によるUWB信号は，搬送波を用いた変調を行わないため，原理的にはベースバンド伝送となり，**図13(a)**に示すパルス信号波形ごとに信号がもつ周波数成分は**図13(b)**のように，0Hz(直流)からGHzまでの広帯域スペクトルをもつことになり，UWBのための周波数割り当てや既存の無線システムとの周波数共用などの電波法上の問題も解決しなければならない．

4 UWBの応用

近年，モバイル情報通信システムにおける大容量化，高信頼化，高品質化に対する要求は専門家のみならず，一般にまで広くブームとなり，そのインフラストラクチャを基にした多様なサービスがビジネス化している．すでに，広帯域無線通信システムとして，Wideband CDMAによるIMT-2000および，その下りリンクのbroadband streamingにHDRが導入され，無線LANにおいても2.4GHz帯のSS(Spread Spectrum)方式によるIEEE801.11bやslow FH(Frequency Hopping)によるBluetooth，さらに5.2GHzのOFDM(Orthogonal Frequency Division Multiplexing)によるHyperLAN2やIEEE802.11aおよびその2.4GHz版といえるIEEE802.11gなどが研究開発され，商用化されている．

これらの方式では広帯域の変調方式を用い，高速無線伝送を可能としている．一方，搬送波を行わずに非常に広い帯域を占めるインパルス信号で無線伝送するUWB方式が，高周波デバイス，信号処理技術の研究開発により実現性を高め，その理論的可能性に対して先端研究者の間で注目を集めている．

UWB方式は先に述べたように，占有帯域幅は非常に広くなり，スペクトル電力密度は非常に小さくなるため，通常のスペクトル拡散通信と同様で，秘話性・秘匿性に優れ，ほかの狭帯域通信に与える影響は小さい．さらに，パルスの時間幅が非常に小さいため，マルチパスを細かく分解でき，RAKE受信が可能となるので，マルチパスにも強い．さらに，超広帯域に周波数スペクトルが拡散されるので，その特徴が顕著に強調される．

これらの特徴を活かした応用が検討されている．とくに，我が国では，UWBの高速伝送が可能で電波法の送信電力の制限などを考慮すると，現在，Bluetoothなどの技術が用いられている近距離ワイヤレスアドホックネットワークの市場に，Bluetoothよりはるかに高速・ブロードバンドなワイヤレス通信を可能にし，センシング機能などをもつ技術として，UWBがにわかに注目されている．

おもな応用として考えられているものは，室内通信，高速無線LAN，ホームネットワーク，コードレス電話，セキュリティセンサ，位置測定器，レーダなどである．とくに，

〔図14〕UWBシステムの応用ターゲット

〔表1〕米国FCCのUWBの規制緩和概要

1998/08	FCC(連邦通信委員会)がUWBを規制緩和するための意見募集(NOI : Notice of Inquiry)を開始
2000/05	FCCがUWBの規制緩和に関する提案書(NPRM : Notice of Proposed Rulemaking)を提出
2001/01	NTIA(National Telecommunication and Information Administration)がGPS以外のFederal Government SystemsへのUWBシステムの影響に関する試験結果を公表
2001/02	NTIAがGPS受信機へもUWBシステムの影響に関する試験結果を公表
2002/02	FCCがUWBの利用に踏み切る(First Report & Order)

(「控えめな」規制緩和 —— 今後さらに規制の緩和を検討)

1)通信

- データ速度:低速(数十kbps)〜超高速(数百Mbps)
- 通信範囲:数m程度(〜5m)
- “The 3rd generation”Bluetooth?
  ⇒無線PAN(Personal Area Network)へ
- IEEE 802.15〔無線PAN(Personal Area Network)〕
- TG3(High Rate, 20Mbps程度まで)での規格を高速化したもの
- おもに家庭内でのデータ伝送
- 近距離での100Mbps以上の無線伝送
- Wireless USB(Universal serial bus)2.0(Intel)
- Data rate : 480Mbps(USB 2.0)
- 米XtremeSpectrum Inc.とTime Domain Corp.がUWB技術の提案を行っている

2)レーダ,センサ

- 軍事用途
- 警察・消防向け(スルーウォールセンサなど)
- 高精度測距(衝突防止センサなど)

などが考えられている.

UWBのこうした応用のターゲットを図示すると,**図14**のように,100Mbps程度の高速ワイヤレスシステムで消費電力をIEEE 802.11a.b.gなどに比べて,十分低減でき,バッテリ駆動の携帯システムや,Bluetoothの応用領域がねらわれる.

5 UWBの法制化

UWBの商用化に関する関心が高まるなか,電波法上の問題がクローズアップされてきている.いうまでもなく,UWBの特徴を活かして応用を展開するためには,GHzオーダの広帯域をどこの周波数帯で利用してよいのか,また,その技術条件は何を考慮して,どのように決めればよいのかなど,多くの課題が残されている.

5.1 米国FCCによるUWB信号に関する規制

UWBの商用化に積極的な米国の状況を,**表1**にまとめる.

2002年2月14日に米国FCCが行ったUWB商用利用承認(詳細は次のURL参照.`http://www.watch.impress.co.jp/internet/www/article/2002/0215/uwb.htm`)に関連して,国内外で商用化に向けた電波法の改正などの検討が進みつつある.まず,米国におけるこれまでの状況は,次のように整理できる.

<UWB規制に関するFCCのこれまでの状況>

① 組織的な許可帯域外の放射は許可されていないが,FCCは無免許システムを基本としたUWBの商用化を許可する方針で検討を進めている.そのためには,組織的な放射に関するFCCの規制Part 15を放棄する必要がある.

② FCC承諾が保留されてきた(part 15 of ET Docket No. 98-153)

現状のRFチャネル割り当て制限は,今後はもはや適用されない見込みである.一方,FCCはGPSなどの既存システムを有害な干渉から保護するための方策を検討中である.新規制を適用する前に,試験や解析のための十分な機会を設定することを明示してきた.

③ こうした試みは,周波数資源の有効利用に大きな貢献を果たすとともに商用UWBシステムの本格導入に大きな牽引力となる.

米国FCCが定める規制Part.5について要約すると,次のように整理できる.

<FCC Part.15規制>

① FCC Part 15制限の原則.微弱電波設備:ライセンス不要.周波数帯ごとの放射電力制限

② FCCのUWBに関する規制案

(a) Communication and Measurement Systems
周波数帯: 3.1GHz〜10.6GHz
IndoorとOutdoorで条件が異なる

(Indoor)　全帯域：－41.3dBm/MHz以下で，室内の場合は Peer-to-peer 型の通信に限定

(Outdoor)　帯域ごとに制限緩和

(b) Imaging Systems

地中探査用レーダ，壁の内部探査用，医療用，セキュリティ用途など(利用者に制限あり)

周波数帯：960MHz以下，1.99GHzまたは3.1GHz～10.6GHz UWB Emission制限

(c) Vehicular Radar Systems

自動車の衝突防止用レーダなど

周波数帯：中心周波数が24.075GHz以上，22GHzと29GHz帯域

③ FCCの無免許組織的電力放射の制限

- f < 960MHz ： 12nW/MHz(－49.3dBm/MHz)
- f > 960MHz ： 75nW/MHz(－41.3dBm/MHz)

④ FCCの非組織的電力放射の制限

- Class Aデバイス：商業，工業，ビジネス市場
  Class A制限：f < 960MHz ： 147nW/MHz (－38.3dBm/MHz)
- Class Bデバイス ：家庭，住居市場
  Class B制限 ：無免許組織的電力放射制限と同じ，

たとえば2.4GHz帯で，1GHzの帯域幅で，UWB送信機から送信できる最大送信電力は－11.3dBmを越えられない．ピーク電力と密度電力の制限以外に考慮すべき重要なパラメータとしては，電界強度，デューティ・サイクル，パルス繰り返し周波数，変調方式，アンテナ利得などを考慮する必要がある．

欧州のETSIにおける基本的な規制は，FCCと同等である．

＊　　＊

UWBの商用化に関するFCCのこのような議論のなかで，もっとも議論が行われている問題は，既存の無線応用システムに対してUWBシステムが与える干渉問題である．この点については，次のように検討が進んでいる．

<UWBが干渉を避けるべき周波数帯域>

① 基本要件

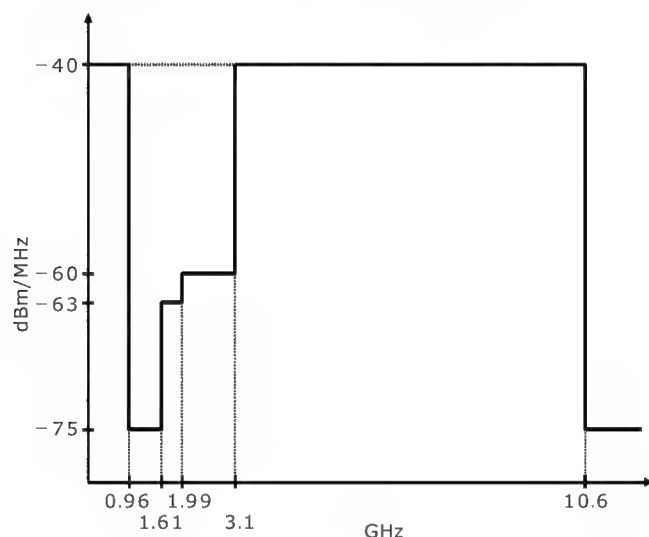
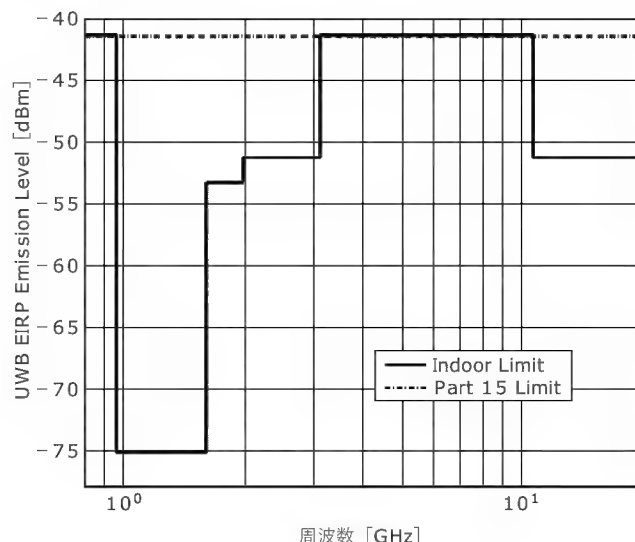
- UWBシステムとともに雑音に関する制限も同時に制度化する必要がある
- 既存の従来からのシステムと共存を考慮する必要があり，とくに，低電力の干渉に対しても敏感なシステムに対する影響を考慮すべきである
- とくに電波天文，GPS，UMTS，航空システムなどのシステムに対しては，周波数共用を禁止するか，十二分に制限する必要がある
- 運用が制限されている65帯域がある

② GPSへの干渉

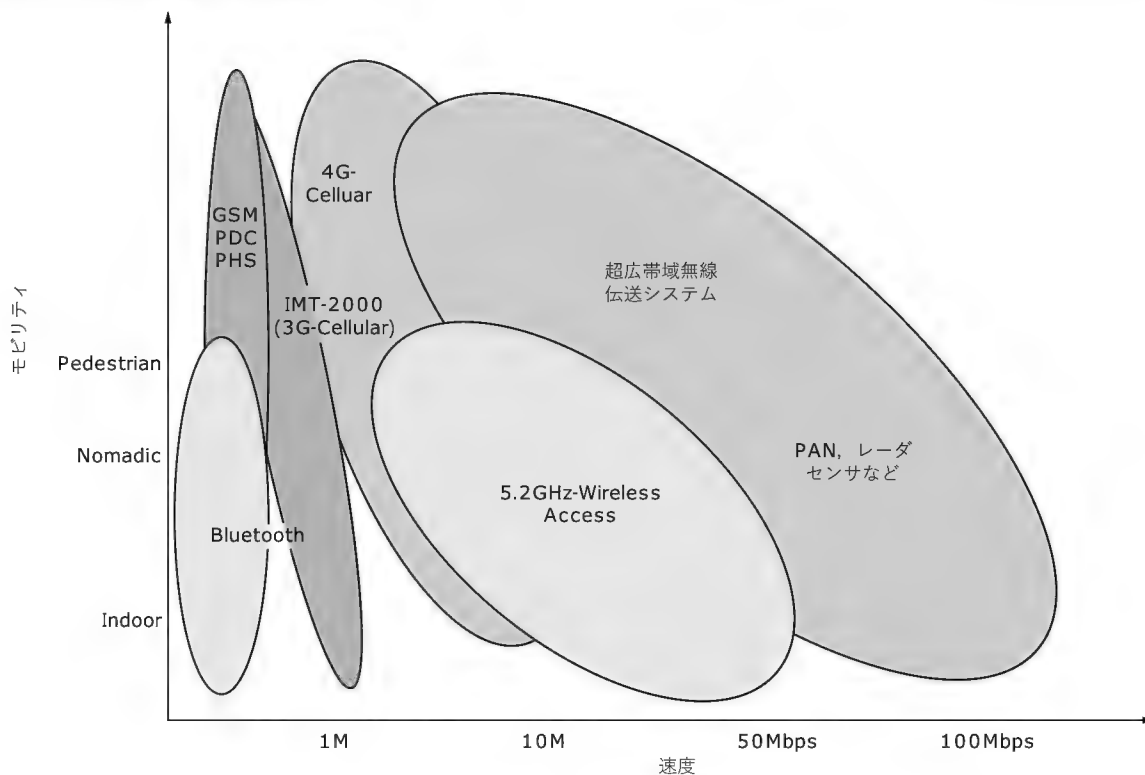
- 衛星からの地上受信信号電力はおおむね：－134dBm(10～16W)
- 典型的なGPS受信感度：－135dBm@1575.42MHz
- UWB信号による干渉は，30～100mの近距離で問題となる
- したがって，UWB信号の帯域を厳しく制限するか，干渉抑圧機能をもつUWB送信機の改良が必要

③ 電波天文への干渉

- 電波天文は，宇宙からのきわめて微弱な電気信号を頼りに天文現象を観測するため，従来からマイクロ波，ミリ波帯の無線通信システムが急増する中で，観測に与える干渉が顕著となり，国際機関URSIのGeneral Assemblyなどでもその対策が議題となっている
- UWB信号はとくに広帯域にわたる尖塔電力の高いパルス信号であるために，電波天文に与える干渉が危惧されている

〔図15〕米国FCCのUWB信号に対する放射電力規制(GPSやセルラーの1～2GHz帯域に対してとくに制限が厳しい)

(a) 屋内伝搬の規制

(b) 屋外伝搬の規制

〔図16〕**UWBシステムの位置付け**(速度対モビリティ)

*　　　　　*

このような状況を考慮し，FCCは現在，UWB用の周波数帯や関連する技術条件を検討し，UWB商用化を支援する視点で，具体案を策定中である．その一端を下記に示す．

<FCCのUWBに関する規制案>

① Communication and Measurement Systems
- 周波数帯：3.1GHz～10.6GHz〔**図15(a)**〕
- 室内，Peer-to-peer型の通信に限定〔**図15(b)**〕

② Imaging Systems
- 地中探査用レーダ，壁の内部探査用，医療用，セキュリティ用途など(利用者に制限あり)
- 960MHz以下，1.99GHzまたは3.1GHz～10.6GHz

③ Vehicular Radar Systems
- 自動車の衝突防止用レーダなど
- 中心周波数が24.075GHz以上

このFCCの活動と連動して，米国の業界団体NTIAや他の政府関係機関などでは，見解を次のように示している．

① NTIAにおけるテスト結果と見解
- (米)政府関係のシステムへの影響
- 400MHz～6.0GHz帯を実験対象
- 3.1GHz以下でのUWBシステムの運用には否定的
- GPS受信機への影響
- 特別なRecommendationはなし

② 米政府関係機関の見解
- 米国防省は4.2GHz帯より高い周波数帯でのUWBの商用利用を認める妥協案
- FAA，NASAは依然として6GHz帯以下における商用UWB無線システムの運用に難色

5.2 独立行政法人通信総合研究所のUWBプロジェクトと産学官コンソーシアム

一方，我が国では，こうした米国主導のUWB商用化に対して，産業界の要請ををふまえ，各種の動きが見られるが，統一的なものは少なかった．これに対し，独立行政法人通信総合研究所(CRL：Communication Research Laboratory)においては，UWBに特化したR&Dグループの準備が2002年5月から進められ，8月にUWBプロジェクトを遂行するUWB結集型特別グループが発足し，マイクロ波からミリ波にいたるUWBの研究開発から技術基準策定などを包括的に遂行し，CRLを中心として横須賀リサーチパーク：YRPにUWB産学官コンソーシアムが結成され，UWBに関する電波制度における国際協調や我が国のUWB商品化に関する産業界への貢献をおもな目的として，産学官連携による共同研究開発が進められている．

UWBシステムの位置付けを**図16**に，システム間の比較を**表2**に示す．

1）UWB産学官コンソーシアムの目的
- 超広帯域ワイヤレスアクセスシステムの研究開発
- テストベッドを用いたマイクロ波帯システム(960MHz，3.1GHz～10.6GHz帯，22GHz～29GHz帯)の実験による検証実証

〔表2〕システム間の比較

項　目	超広帯域無線システムの目標	Bluetooth	Bluetooth Ver.2	5.2GHz 移動アクセス	60GHz 帯免許不要システム（※2）
ユーザー速度	100Mbps 以上	最大 721kbps	2Mbps 程度（※1）	最大 54Mbps	Home-link で 1.6Gbps 程度
通信距離	10 m程度	10～100 m程度	10～100 m程度	数 100 m	10 m程度
短　所	短距離通信	低データ速度		消費電力	ハードウェアコスト
特　徴	●低消費電力 ● Ad-Hoc ●低価格 ●高速伝送 ●測距・測位	● Ad-Hoc ●低価格	● Ad-Hoc ●低価格	屋内利用に限定	高速伝送

※1：http://www.ericsson.co.jp/products/bluetooth_ip/faq/faq_02c.html#33
※2：電技審 60GHz 帯無線設備委員会資料の一例

- ●未利用周波数帯（準ミリ波～ミリ波帯）の研究開発
- ●高速データ伝送（100Mbp 以上）を達成し得る，低コスト化送受信モジュール，通信方式の確立
- ●情報技審/ARIB などにおける標準化への寄与

2）UWB コンソーシアムにおける主要研究課題

- ●周波数共用化技術
- ●超広帯域アドホック通信方式
- ●高速（100Mbps 以上）伝送技術
- ●超広帯域マイクロ波・ミリ波デバイス技術
- ●超広帯域マイクロ波・ミリ波アンテナ技術
- ●電波伝搬特性の解明およびモデル化
- ●干渉波抑圧・除去方式
- ●高速パルス信号処理技術（RF 帯，BB 帯）
- ●位置測定方式

3）UWB コンソーシアムの共同研究の進め方

- ●契約形態
  (a) YRP 研究開発協議会において参加企業を募る
  (b) 企業は CRL と 1：N の共同研究契約を締結
- ●マイクロ波方式に関する共同研究
  (a) 平成 14 年 11 月より平成 16 年 3 月までの約 1.5 年間
- ●ミリ波方式に関する共同研究
  (a) 平成 14 年 11 月より平成 18 年 3 月までの約 3.5 年間

6 UWBの研究開発課題とまとめ

本章では，インパルス無線による UWB の原理，特徴，応用，法規制などについて紹介してきた．UWB は未だ研究段階にあり，商用化にのためには解決すべき課題が山積している．その研究課題は，先に述べた UWB の問題点を解決することに集約される．技術的な問題点については，多くの研究者が新たに取り組むべき問題として注目し，UWB に特化した国際会議 UWBST 02（http://www.uwbst2002.com/）も 2002 年 5 月に米国 Baltimore で開催され，米国ばかりでなく，欧州，日本から先進的な研究成果が報告されている．

筆者らは，比較的早い時期から UWB の研究に着手し，研究契約などの問題もあるが，まとまった研究成果を最近外部に報告してきている．その一つとして，従来のスペクトル拡散方式である直接拡散（DS）方式や周波数ホッピング方式（FH）などとの関係や比較についても紹介してきた．

＜UWB と SS との比較＞

DS-SS と UWB を占有帯域幅，伝送速度を同一にした場合に誤り率特性を比較しよう．

1）DS-SS の *BER* 理論式

$$SNR=\frac{1}{\frac{K-1}{3N}-\frac{N_0}{2E_b}}$$

N：1 ビット時間中のチップ数

K：ユーザー数

$$BER=\frac{1}{2}erfc\left(\sqrt{\frac{SNR}{2}}\right)$$

2）FH-SS の *BER* 理論式

$$BER=\frac{1}{2}\left(1-\frac{1}{k}\right)^{M-1}erfc\left(\sqrt{\frac{S}{2N}}\right)+\frac{1}{2}\sum_{i=1}^{M-1}\left(\frac{1}{k}\right)^{i}{}_{M-1}C_i\,erfc\left(\sqrt{\frac{S}{2N+S_i}}\right)$$

K：ホッピング周波数の個数（850）

M：ユーザー数

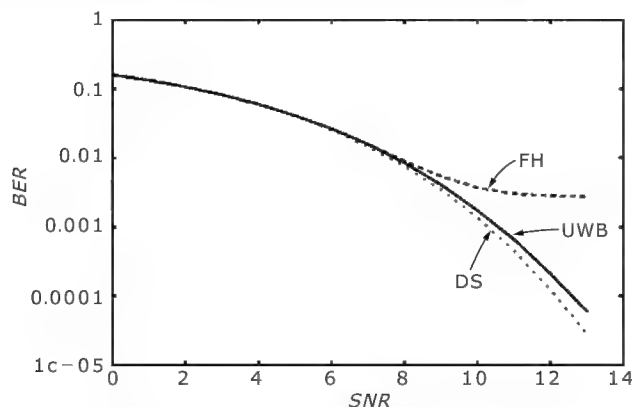
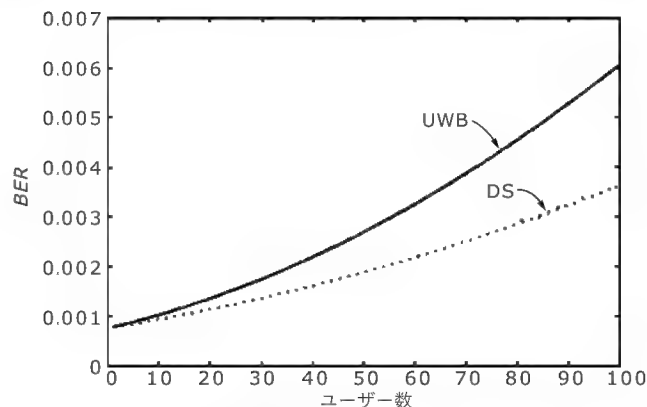
S：信号エネルギ

N：雑音エネルギ

$erfc$：ガウスの誤り関数

伝送速度 3.125Mbps，ユーザー数 30，周波数帯域幅（ピーク値から 3dB 下がる帯域幅）3.2GHz とした場合の DS-SS，FH-SS と UWB の *BER* 特性を**図 17（a）**に示す．これより，もし従来の SS 方式で UWB 方式と同程度の帯域幅に信号スペクトルを拡散できれば，DS も FH も UWB とビット誤り率（*BER*）特性はあまりかわらないことがわかる．しかし，FH はユーザー数が増えるとユーザーごとに各時点で用いる周波数が衝突（ヒット）しやすくなるため，信号対雑音平均電力比（*SNR*）を大きくしても，誤り率（*BER*）は改善されなくなる．

図 17（b）に，ユーザー数が増加した場合の UWB 方式と DS 方式の *BER* を示している．これから，UWB 方式は一種のタイム

〔図17〕UWBシステムと従来のSSシステム(DS，FH)の性能比較

(a) *S/N*比対*BER*(ビット誤り率)特性(伝送速度3.125Mbps，ユーザー数30，周波数帯域幅3.2GHz)

(b) 同時利用ユーザー数対*BER*(ビット誤り率)特性(伝送速度3.125Mbps，SNR＝30dB，周波数帯域幅3.2GHz)

ホッピング(TH)方式であるため，FH方式と同様にユーザー数が増加すると，ユーザー間のパルスの衝突が増加するため，*SNR*を改善しても*BER*は改善されなくなる．

このことから，UWB方式についても周波数利用効率を向上させるには，衝突を軽減する各種の技術が必要となる．しかし当初は，理論的限界ほどのユーザー数をサービスするような応用は現時点では想定されていない．また，DS方式やFH方式でUWB方式と同等の超広帯域へ拡散することは，経済的には困難である．

これらの理解に基づき，筆者らの研究成果として，UWBの問題点を解決する方式，とくに，多元接続における干渉問題の解消法として，UWBマルチユーザー受信法，マルチパス環境下での対処法として，RAKE受信法，高効率伝送のための多値化，多元接続法などについて研究開発を行ってきた．今後はさらに，商用化に必要な技術基準や技術条件適合証明のための測定法などの研究も必要と考える．

参考文献

1) Moe Z. Win, Robert A. Scholtz, "Ultra-Wide Bandwidth Time-Hopping Spread-Spectrum Impulse Radio for Wireless Multiple-Access Communications", *IEEE Transactions on Communications*, Vol.48, No.4, pp679-691, April 2000
2) J.M.Cramer, R.A.Scholtz, M.Z.Win, "On the analysis of UWB communication channels", pp.1191-1195, *Proc. IEEE Military Communications Conf.* 1999
3) F.Ramirez-Mireles, R.A.Scholtz, "Multiple access performance limits with time -hopping and pulse position modulation", pp.529-533 *Proc. IEEE Military Communication Conf.* 1998
4) 江島，梅林，水谷，河野，「Impulse Radio(UWB)の多値化とマルチユーザー用干渉除去方式の一検討」，『信学技報』，SST2001-16，April 2001
5) 丸林 元，中川 正雄，河野 隆二共著，『スペクトル拡散とその応用』，信学会，May 2000

こうの・りゅうじ kohno@ynu.ac.jp 横浜国立大学大学院

x86CPUだけでもマスタしたい

開発技術者のためのアセンブラ入門

第⑮回　2進演算命令の加減算

大貫広幸

今回と次回の2回で，x86系の32ビットCPUで使用できる算術命令について説明します．

2進算術命令

算術命令は，算術演算(arithmetic operation)を行う命令で，加減乗除などの四則演算などのことをいいます．

算術命令は，分類上「2進算術命令」と「10進算術命令」に分けられます．

2進算術命令は，2進数で表される整数値に対する演算命令です．そして，10進算術命令は，BCDで表される10進数の整数値に対する演算命令となります．

2進算術命令は，汎用レジスタあるいはメモリ上にある2進数で表される整数値に対して加減乗除の四則演算を行うものです．

扱える2進整数は，386以降の32ビットCPUでは，8ビット長のバイト整数，16ビット長のワード整数，そして32ビット長のダブルワード整数の3種類です．

符号については，符号付き整数と符号なし整数の両方を扱うことができます．そして，符号付き整数の負数は，2の補数で値を表します．

2進算術命令では，機械語命令を実行すると，演算結果にしたがいステータスフラグ(OF, SF, ZF, AF, PF, CF)が設定されます．そのため，2進算術命令実行後，ステータスフラグを見ることで，演算結果の状態を知ることができます．

これは，先に述べたCMOVcc命令や後の回で説明するSETcc命令，そして条件ジャンプ命令などの判断条件として使用されます．

2進算術命令には，加算関係の命令としてADD，ADC，INCの3命令，減算関係の命令としてSUB，SBB，CMP，NEG，DECの5命令，乗算関係の命令としてMUL，IMULの2命令，そして除算関係の命令としてDIV，IDIVの2命令，合計12個の命令があります．

今回はこのうち，**表1**に示す2進算術命令の加減算に関する8命令について説明します．

〔表1〕x86系の32ビットCPUで使用できる2進加減算命令

分類	インストラクション名	動　作	影響を受けるフラグ					
			OF	SF	ZF	AF	PF	CF
2進算術命令	ADD	Add 2進数による加算 DEST ← DEST + SOU	*	*	*	*	*	*
	ADC	Add with Carry 2進数によるキャリを含む加算 DEST ← DEST + SOU + CF	*	*	*	*	*	*
	INC	Increment 2進数による1の加算 DEST ← DEST + 1	*	*	*	*	*	·
	SUB	Subtract 2進数による減算 DEST ← DEST − SOU	*	*	*	*	*	*
	SBB	Subtract with Borrow 2進数によるボローを含む減算 DEST ← DEST −(SOU + CF)	*	*	*	*	*	*
	CMP	Compare 2進数による二つのオペランドの大小比較　SOU1 − SOU2	*	*	*	*	*	*
	NEG	Negate オペランドの2進数の値を2の補数した値で置き換える DEST ← − DEST	*	*	*	*	*	*
	DEC	Decrement 2進数による1の減算 DEST ← DEST − 1	*	*	*	*	*	·

▶表中のDESTはdestination(先)，SOUはsource(元)
▶表中の影響を受けるフラグの記号は次の状態を表す
・＝変化ない　＊＝結果にしたがい変化する

2進算術命令の加算命令

加算関係の命令としてADD，ADC，INCの3命令があります．

● ADD命令

ADD命令は，オペランドの転送先をDEST，転送元をSOUで表すと，

DEST ← DEST + SOU

という加算を行います．

転送先(DEST)には汎用レジスタやメモリ上の値が指定できます．そして，転送元(SOU)には汎用レジスタやメモリ上の値，イミディエイトが指定できます．

ただし，メモリ上の値は，二つ指定できないため，一つのオペランドでメモリ上の値を指定した場合は，もう一方

のオペランドは，汎用レジスタかイミディエイトとなります．

そのため，演算のバリエーションは，

- 汎用レジスタ ← 汎用レジスタ＋汎用レジスタ
- 汎用レジスタ ← 汎用レジスタ＋メモリ
- メモリ ← メモリ＋汎用レジスタ
- 汎用レジスタ ← 汎用レジスタ＋イミディエイト
- メモリ ← メモリ＋イミディエイト

となります．

ADD 命令実行後，ステータスフラグ(OF，SF，ZF，AF，PF，CF)は，演算結果にしたがい設定されます．ADD 命令自体は，符号なし，符号付きの両方の演算に使用します．

そのため，ステータスフラグも符号なし，符号付きの両方の演算の結果を表すことになります．

符号に関係のないフラグとしては，結果がゼロになった場合の ZF，結果が偶数パリティになった場合の PF，加算の結果，最上位ビットからの桁上がりの発生を示す CF と，ビット 3 からの桁上がりの発生を示す AF があります．

符号なしに関係する結果としては，CF が符号なし演算の結果，オーバフローしたことを示すフラグとしても使用されます．

符号付きに関係する結果としては，OF がオーバフローを表し，SF が負数を表しています．

実際の MASM での ADD 命令の記述例を**リスト 1**(次頁)に，gas での ADD 命令の記述例を**リスト 2**(p.117)に示します．

● ADC 命令

ADC 命令は，オペランドの転送先を DEST，転送元を SOU で表すと，

DEST ← DEST ＋ SOU ＋ CF

という加算を行います．CF は，ステータスフラグの CF です．

ADC 命令のオペランドの指定は，ADD 命令とまったく同じです．そのため，演算のバリエーションは，

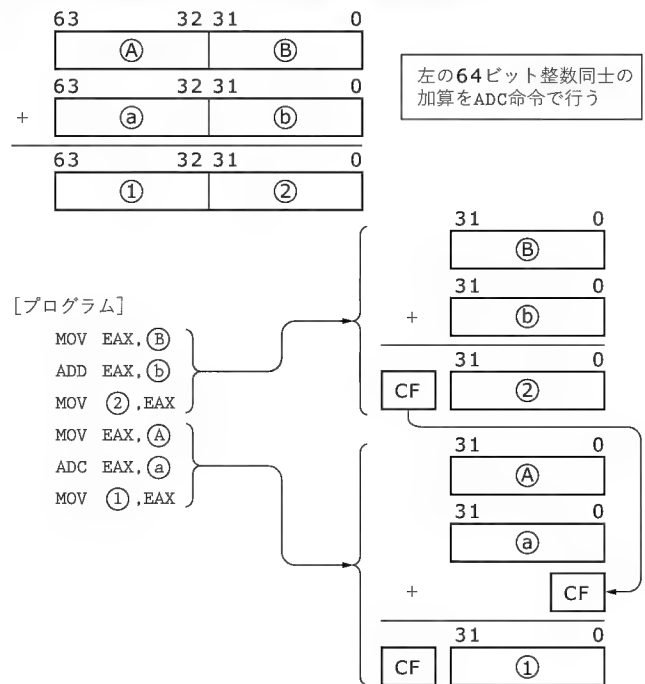
- 汎用レジスタ ← 汎用レジスタ＋汎用レジスタ＋ CF
- 汎用レジスタ ← 汎用レジスタ＋メモリ＋ CF
- メモリ ← メモリ＋汎用レジスタ＋ CF
- 汎用レジスタ ← 汎用レジスタ＋イミディエイト＋ CF
- メモリ ← メモリ＋イミディエイト＋ CF

となります．

ADC 命令実行後，ステータスフラグ(OF，SF，ZF，AF，PF，CF)は，演算結果にしたがい設定されます．ステータスフラグの設定のされ方も，ADD 命令とまったく同じになっています．

実際の MASM での ADC 命令の記述例を**リスト 1**，gas での ADC 命令の記述例を**リスト 2** に示します．

ADC 命令は，32 ビット長のダブルワード整数よりビット数が多い整数値の加算に使用します．たとえば，**図 1** のような 64 ビット長のクワッドワードで表される整数の加算なら，まず下位のダブルワード同士を ADD 命令で加算し，次に上位のダブルワード同士を ADC 命令で加算することで，64 ビット長の加算が行えます．

〔図 1〕ADC 命令による 64 ビット整数同士の加算の例

このように，ADC 命令を必要な回数使うことで，ビット数が多い整数値の加算も行うことが可能となります．

● INC 命令

INC 命令のオペランドは転送先一つで，オペランドで指定された転送先の汎用レジスタあるいはメモリ上の値を＋ 1 する命令です．

つまり，オペランドの転送元を DEST で表すと，

DEST ← DEST ＋ 1

という加算を行います．

INC 命令を実行すると，CF を除くステータスフラグ(OF，SF，ZF，AF，PF)は，演算結果にしたがい設定されます．つまり，INC 命令では CF は変化しません．そのため，INC 命令の実行により最上位ビットからの桁上がりが発生した場合でも，その桁上がりは CF には設定されません．

実際の MASM での INC 命令の記述例を**リスト 1** に，gas での INC 命令の記述例を**リスト 2** に示します．

2 進算術命令の減算命令

減算関係の命令として SUB，SBB，CMP，NEG，DEC の五つの命令があります．

● SUB 命令

SUB 命令は，オペランドの転送先を DEST，転送元を SOU で表すと，

DEST ← DEST − SOU

という減算を行います．

転送先(DEST)には汎用レジスタやメモリ上の値が指定でき

〔リスト 1〕MASM の ADD，ADC，INC 命令の記述例

```
                                .586
                                .model flat

00000000                        .data

00000000 01                     dtByte  db      1
00000001 0002                   dtWord  dw      2
00000003 00000004               dtDWord dd      4

00000000                        .code
                                ;**************************************
                                ;       ADD
                                ;**************************************
                                ; 汎用レジスタ←汎用レジスタ + 汎用レジスタ
00000000  02 F8                         add     bh,al
00000002  66| 03 DF                     add     bx,di
00000005  03 F9                         add     edi,ecx
                                ; 汎用レジスタ←汎用レジスタ + メモリ
00000007  02 1D 00000000 R              add     bl,dtByte
0000000D  66| 03 35                     add     si,dtWord
          00000001 R
00000014  03 0D 00000003 R              add     ecx,dtDWord
                                ; メモリ←メモリ + 汎用レジスタ
0000001A  00 35 00000000 R              add     dtByte,dh
00000020  66| 01 0D                     add     dtWord,cx
          00000001 R
00000027  01 1D 00000003 R              add     dtDWord,ebx
                                ; アキュムレータ←アキュムレータ + イミディエイト
0000002D  04 12                         add     al,12h
0000002F  66| 05 1234                   add     ax,1234h
00000033  05 12345678                   add     eax,12345678h
                                ; 汎用レジスタ←汎用レジスタ + イミディエイト
00000038  80 C7 12                      add     bh,12h
0000003B  66| 81 C1 1234                add     cx,1234h
00000040  81 C7 12345678                add     edi,12345678h
                                ; メモリ←メモリ + イミディエイト
00000046  80 05 00000000 R              add     dtByte,12h
          12
0000004D  66| 81 05                     add     dtWord,1234h
          00000001 R
          1234
00000056  81 05 00000003 R              add     dtDWord,12345678h
          12345678

                                ;**************************************
                                ;       ADC
                                ;**************************************
                                ; 汎用レジスタ←汎用レジスタ + 汎用レジスタ + CF
00000060  12 F8                         adc     bh,al
00000062  66| 13 DF                     adc     bx,di
00000065  13 F9                         adc     edi,ecx
                                ; 汎用レジスタ←汎用レジスタ + メモリ + CF
00000067  12 1D 00000000 R              adc     bl,dtByte
0000006D  66| 13 35                     adc     si,dtWord
          00000001 R
00000074  13 0D 00000003 R              adc     ecx,dtDWord
                                ; メモリ←メモリ + 汎用レジスタ + CF
0000007A  10 35 00000000 R              adc     dtByte,dh
00000080  66| 11 0D                     adc     dtWord,cx
          00000001 R
00000087  11 1D 00000003 R              adc     dtDWord,ebx
                                ; アキュムレータ←アキュムレータ + イミディエイト + CF
0000008D  14 12                         adc     al,12h
0000008F  66| 15 1234                   adc     ax,1234h
00000093  15 12345678                   adc     eax,12345678h
                                ; 汎用レジスタ←汎用レジスタ + イミディエイト + CF
00000098  80 D7 12                      adc     bh,12h
0000009B  66| 81 D1 1234                adc     cx,1234h
000000A0  81 D7 12345678                adc     edi,12345678h
                                ; メモリ←メモリ + イミディエイト + CF
000000A6  80 15 00000000 R              adc     dtByte,12h
          12
000000AD  66| 81 15                     adc     dtWord,1234h
          00000001 R
          1234
000000B6  81 15 00000003 R              adc     dtDWord,12345678h
          12345678

                                ;**************************************
                                ;       INC
                                ;**************************************
000000C0  FE C1                         inc     cl
000000C2  66| 41                        inc     cx
000000C4  41                            inc     ecx
000000C5  FE 05 00000000 R              inc     dtByte
000000CB  66| FF 05                     inc     dtWord
          00000001 R
000000D2  FF 05 00000003 R              inc     dtDWord

                                end
```

アキュムレータに対するイミディエイトの加算は専用の機械語命令をもっているので，他の汎用レジスタに対する加算より多少機械語のバイト数が短くなる

〔リスト2〕gasのADD，ADC，INC命令の記述例

```
 1                      .data
 2
 3  0000 01             dtByte:    .byte  1
 4  0001 0200           dtWord:    .word  2
 5  0003 04000000       dtDWord:   .long  4
 6
 7                      .text
 8                      #**************************************
 9                      #        ADD
10                      #**************************************
11                      # 汎用レジスタ + 汎用レジスタ → 汎用レジスタ
12  0000 00C7                   addb    %al,%bh
13  0002 6601FB                 addw    %di,%bx
14  0005 01CF                   addl    %ecx,%edi
15                      # 汎用レジスタ + メモリ → 汎用レジスタ
16  0007 021D0000               addb    dtByte,%bl
16       0000
17  000d 66033501               addw    dtWord,%si
17       000000
18  0014 030D0300               addl    dtDWord,%ecx
18       0000
19                      # メモリ + 汎用レジスタ → メモリ
20  001a 00350000               addb    %dh,dtByte
20       0000
21  0020 66010D01               addw    %cx,dtWord
21       000000
22  0027 011D0300               addl    %ebx,dtDWord
22       0000
23                      # アキュムレータ + イミディエイト → アキュムレータ
24  002d 0412                   addb    $0x12,%al
25  002f 66053412               addw    $0x1234,%ax
26  0033 05785634               addl    $0x12345678,%eax
26       12
27                      # 汎用レジスタ + イミディエイト → 汎用レジスタ
28  0038 80C712                 addb    $0x12,%bh
29  003b 6681C134               addw    $0x1234,%cx
29       12
30  0040 81C77856               addl    $0x12345678,%edi
30       3412
31                      # メモリ + イミディエイト → メモリ
32  0046 80050000               addb    $0x12,dtByte
32       000012
33  004d 66810501               addw    $0x1234,dtWord
33       00000034
33       12
34  0056 81050300               addl    $0x12345678,dtDWord
34       00007856
34       3412
35                      #**************************************
36                      #        ADC
37                      #**************************************
38                      # 汎用レジスタ + 汎用レジスタ + CF → 汎用レジスタ
39  0060 10C7                   adcb    %al,%bh
40  0062 6611FB                 adcw    %di,%bx
41  0065 11CF                   adcl    %ecx,%edi
42                      # 汎用レジスタ + メモリ + CF → 汎用レジスタ
43  0067 121D0000               adcb    dtByte,%bl
43       0000
44  006d 66133501               adcw    dtWord,%si
44       000000
45  0074 130D0300               adcl    dtDWord,%ecx
45       0000
46                      # メモリ + 汎用レジスタ + CF → メモリ
47  007a 10350000               adcb    %dh,dtByte
47       0000
48  0080 66110D01               adcw    %cx,dtWord
48       000000
49  0087 111D0300               adcl    %ebx,dtDWord
49       0000
50                      # アキュムレータ + イミディエイト + CF → アキュムレータ
51  008d 1412                   adcb    $0x12,%al
52  008f 66153412               adcw    $0x1234,%ax
53  0093 15785634               adcl    $0x12345678,%eax
53       12
54                      # 汎用レジスタ + イミディエイト + CF → 汎用レジスタ
55  0098 80D712                 adcb    $0x12,%bh
56  009b 6681D134               adcw    $0x1234,%cx
56       12
57  00a0 81D77856               adcl    $0x12345678,%edi
57       3412
58                      # メモリ + イミディエイト + CF → メモリ
59  00a6 80150000               adcb    $0x12,dtByte
59       000012
60  00ad 66811501               adcw    $0x1234,dtWord
60       00000034
60       12
61  00b6 81150300               adcl    $0x12345678,dtDWord
61       00007856
61       3412
62                      #**************************************
63                      #        INC
64                      #**************************************
65  00c0 FEC1                   incb    %cl
66  00c2 6641                   incw    %cx
67  00c4 41                     incl    %ecx
68  00c5 FE050000               incb    dtByte
68       0000
69  00cb 66FF0501               incw    dtWord
69       000000
70  00d2 FF050300               incl    dtDWord
70       0000
```

ます．そして，転送元(SOU)には汎用レジスタやメモリ上の値，イミディエイトが指定できます．

ただし，メモリ上の値は，二つ指定できないため，一つのオペランドでメモリ上の値を指定した場合は，もう一方のオペランドは，汎用レジスタかイミディエイトとなります．

そのため，演算のバリエーションは，

- 汎用レジスタ ← 汎用レジスタ－汎用レジスタ
- 汎用レジスタ ← 汎用レジスタ－メモリ
- メモリ ← メモリ－汎用レジスタ
- 汎用レジスタ ← 汎用レジスタ－イミディエイト
- メモリ ← メモリ－イミディエイト

となります．

SUB命令実行後，ステータスフラグ(OF，SF，ZF，AF，PF，CF)は，演算結果にしたがい設定されます．SUB命令自体は，符号なし，符号付きの両方の演算に使用します．

そのため，ステータスフラグも符号なし，符号付きの両方の演算の結果を表すことになります．

符号に関係のないフラグとしては，結果がゼロになった場合のZF，結果が偶数パリティになった場合のPF，減算の結果，最上位ビットの桁借りの発生を示すCFと，ビット3の桁借りの発生を示すAFがあります．

符号なしに関係する結果としては，CFが符号なし演算の結果，オーバフローしたことを示すフラグとしても使用されます．

符号付きに関係する結果としては，OFがオーバフローを表し，SFが負数を表しています．

実際のMASMでのSUB命令の記述例を**リスト3**(次頁)，gasでのSUB命令の記述例を**リスト4**(p.120)に示します．

● SBB命令

SBB命令は，オペランドの転送先をDEST，転送元をSOUで表すと，

DEST ← DEST－(SOU＋CF)

という減算を行います．CFは，ステータスフラグのCFです．

〔リスト3〕MASMのSUB，SBB，CMP，NEG，DEC命令の記述例

```
                                .586
                                .model flat

00000000                        .data

00000000 01                     dtByte  db      1
00000001 0002                   dtWord  dw      2
00000003 00000004               dtDWord dd      4

00000000                        .code
                                ;**************************************
                                ;       SUB
                                ;**************************************
                                ; 汎用レジスタ←汎用レジスタ - 汎用レジスタ
00000000  2A F8                         sub     bh,al
00000002  66| 2B DF                     sub     bx,di
00000005  2B F9                         sub     edi,ecx
                                ; 汎用レジスタ←汎用レジスタ - メモリ
00000007  2A 1D 00000000 R              sub     bl,dtByte
0000000D  66| 2B 35                     sub     si,dtWord
          00000001 R
00000014  2B 0D 00000003 R              sub     ecx,dtDWord
                                ; メモリ←メモリ - 汎用レジスタ
0000001A  28 35 00000000 R              sub     dtByte,dh
00000020  66| 29 0D                     sub     dtWord,cx
          00000001 R
00000027  29 1D 00000003 R              sub     dtDWord,ebx
                                ; アキュムレータ←アキュムレータ - イミディエイト
0000002D  2C 12                         sub     al,12h
0000002F  66| 2D 1234                   sub     ax,1234h
00000033  2D 12345678                   sub     eax,12345678h
                                ; 汎用レジスタ←汎用レジスタ - イミディエイト
00000038  80 EF 12                      sub     bh,12h
0000003B  66| 81 E9 1234                sub     cx,1234h
00000040  81 EF 12345678                sub     edi,12345678h
                                ; メモリ←メモリ - イミディエイト
00000046  80 2D 00000000 R              sub     dtByte,12h
          12
0000004D  66| 81 2D                     sub     dtWord,1234h
          00000001 R
          1234
00000056  81 2D 00000003 R              sub     dtDWord,12345678h
          12345678

                                ;**************************************
                                ;       SBB
                                ;**************************************
                                ; 汎用レジスタ←汎用レジスタ - (汎用レジスタ + CF)
00000060  1A F8                         sbb     bh,al
00000062  66| 1B DF                     sbb     bx,di
00000065  1B F9                         sbb     edi,ecx
                                ; 汎用レジスタ←汎用レジスタ - (メモリ + CF)
00000067  1A 1D 00000000 R              sbb     bl,dtByte
0000006D  66| 1B 35                     sbb     si,dtWord
          00000001 R
00000074  1B 0D 00000003 R              sbb     ecx,dtDWord
                                ; メモリ←メモリ - (汎用レジスタ + CF)
0000007A  18 35 00000000 R              sbb     dtByte,dh
00000080  66| 19 0D                     sbb     dtWord,cx
          00000001 R
00000087  19 1D 00000003 R              sbb     dtDWord,ebx
                                ; アキュムレータ←アキュムレータ - (イミディエイト + CF)
0000008D  1C 12                         sbb     al,12h
0000008F  66| 1D 1234                   sbb     ax,1234h
00000093  1D 12345678                   sbb     eax,12345678h
                                ; 汎用レジスタ←汎用レジスタ - (イミディエイト + CF)
00000098  80 DF 12                      sbb     bh,12h
0000009B  66| 81 D9 1234                sbb     cx,1234h
000000A0  81 DF 12345678                sbb     edi,12345678h
                                ; メモリ←メモリ - (イミディエイト + CF)
000000A6  80 1D 00000000 R              sbb     dtByte,12h
          12
000000AD  66| 81 1D                     sbb     dtWord,1234h
          00000001 R
          1234
000000B6  81 1D 00000003 R              sbb     dtDWord,12345678h
          12345678

                                ;**************************************
                                ;       CMP
                                ;**************************************
                                ; 汎用レジスタ - 汎用レジスタ
000000C0  38 C7                         cmp     bh,al
000000C2  66| 3B DF                     cmp     bx,di
000000C5  3B F9                         cmp     edi,ecx
                                ; 汎用レジスタ - メモリ
000000C7  3A 1D 00000000 R              cmp     bl,dtByte
000000CD  66| 3B 35                     cmp     si,dtWord
          00000001 R
000000D4  3B 0D 00000003 R              cmp     ecx,dtDWord
```

アキュムレータに対するイミディエイトの減算と比較は，専用の機械語命令をもっているので，他の汎用レジスタに対する加算より多少機械語のバイト数が短くなる

〔リスト3〕**MASMのSUB，SBB，CMP，NEG，DEC命令の記述例**（つづき）

```
                                    ; メモリ - 汎用レジスタ
000000DA  38 35 00000000 R                cmp     dtByte,dh
000000E0  66| 39 0D                       cmp     dtWord,cx
          00000001 R
000000E7  39 1D 00000003 R                cmp     dtDWord,ebx
                                    ; アキュムレータ - イミディエイト
000000ED  3C 12                           cmp     al,12h
000000EF  66| 3D 1234                     cmp     ax,1234h
000000F3  3D 12345678                     cmp     eax,12345678h
                                    ; 汎用レジスタ - イミディエイト
000000F8  80 FF 12                        cmp     bh,12h
000000FB  66| 81 F9 1234                  cmp     cx,1234h
00000100  81 FF 12345678                  cmp     edi,12345678h
                                    ; メモリ - イミディエイト
00000106  80 3D 00000000 R                cmp     dtByte,12h
          12
0000010D  66| 81 3D                       cmp     dtWord,1234h
          00000001 R
          1234
00000116  81 3D 00000003 R                cmp     dtDWord,12345678h
          12345678

                                    ;**************************************
                                    ;       NEG
                                    ;**************************************
00000120  F6 D9                           neg     cl
00000122  66| F7 D9                       neg     cx
00000125  F7 D9                           neg     ecx
00000127  F6 1D 00000000 R                neg     dtByte
0000012D  66| F7 1D                       neg     dtWord
          00000001 R
00000134  F7 1D 00000003 R                neg     dtDWord
                                    ;**************************************
                                    ;       DEC
                                    ;**************************************
0000013A  FE C9                           dec     cl
0000013C  66| 49                          dec     cx
0000013E  49                              dec     ecx
0000013F  FE 0D 00000000 R                dec     dtByte
00000145  66| FF 0D                       dec     dtWord
          00000001 R
0000014C  FF 0D 00000003 R                dec     dtDWord

                                    end
```

つまり，SBB命令ではまず転送元(SOU)にCFを加算し，その加算結果を転送先(DEST)から引くという動作をします．この「DEST ← DEST −（SOU + CF）」の計算は，

DEST ← DEST − SOU − CF

と表すこともできます．

SBB命令のオペランドの指定は，SUB命令とまったく同じです．そのため，演算のバリエーションは，

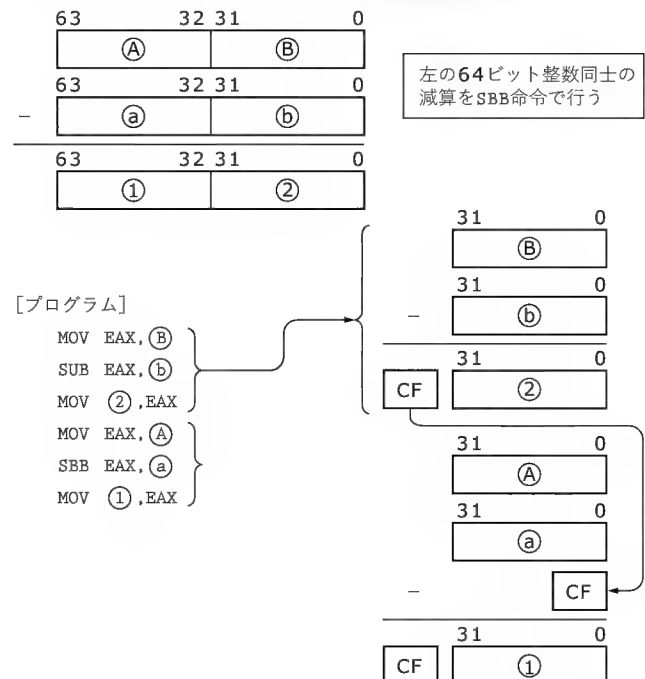
- 汎用レジスタ ← 汎用レジスタ −（汎用レジスタ + CF）
- 汎用レジスタ ← 汎用レジスタ −（メモリ + CF）
- メモリ ← メモリ −（汎用レジスタ + CF）
- 汎用レジスタ ← 汎用レジスタ −（イミディエイト + CF）
- メモリ ← メモリ −（イミディエイト + CF）

となります．

SBB命令実行後，ステータスフラグ(OF，SF，ZF，AF，PF，CF)は，演算結果にしたがい設定されます．ステータスフラグの設定のされ方も，SUB命令とまったく同じになっています．

実際のMASMでのSBB命令の記述例を**リスト3**，gasでのSBB命令の記述例を**リスト4**(p.120)に示します．

SBB命令は，32ビット長のダブルワード整数よりビット数が多い整数値の減算に使用します．たとえば，**図2**のような64ビット長のクワッドワードで表される整数の減算なら，まず下位のダブルワード同士をSUB命令で減算し，次に上位のダブルワ

〔図2〕**SBB命令による64ビット整数同士の減算の例**

〔リスト4〕gasのSUB，SBB，CMP，NEG，DEC命令の記述例

```
  1                       .data
  2
  3 0000 01         dtByte:    .byte  1
  4 0001 0200       dtWord:    .word  2
  5 0003 04000000   dtDWord:   .long  4
  6
  7                 .text
  8                 #*************************************
  9                 #       SUB
 10                 #*************************************
 11                 # 汎用レジスタ - 汎用レジスタ → 汎用レジスタ
 12 0000 28C7               subb    %al,%bh
 13 0002 6629FB             subw    %di,%bx
 14 0005 29CF               subl    %ecx,%edi
 15                 # 汎用レジスタ - メモリ → 汎用レジスタ
 16 0007 2A1D0000           subb    dtByte,%bl
 16      0000
 17 000d 662B3501           subw    dtWord,%si
 17      000000
 18 0014 2B0D0300           subl    dtDWord,%ecx
 18      0000
 19                 # メモリ - 汎用レジスタ → メモリ
 20 001a 28350000           subb    %dh,dtByte
 20      0000
 21 0020 66290D01           subw    %cx,dtWord
 21      000000
 22 0027 291D0300           subl    %ebx,dtDWord
 22      0000
 23                 # アキュムレータ - イミディエイト → アキュムレータ
 24 002d 2C12               subb    $0x12,%al
 25 002f 662D3412           subw    $0x1234,%ax
 26 0033 2D785634           subl    $0x12345678,%eax
 26      12
 27                 # 汎用レジスタ - イミディエイト → 汎用レジスタ
 28 0038 80EF12             subb    $0x12,%bh
 29 003b 6681E934           subw    $0x1234,%cx
 29      12
 30 0040 81EF7856           subl    $0x12345678,%edi
 30      3412
 31                 # メモリ - イミディエイト → メモリ
 32 0046 802D0000           subb    $0x12,dtByte
 32      000012
 33 004d 66812D01           subw    $0x1234,dtWord
 33      00000034
 33      12
 34 0056 812D0300           subl    $0x12345678,dtDWord
 34      00007856
 34      3412
 35               #*************************************
 36               #       SBB
 37               #*************************************
 38               # 汎用レジスタ - (汎用レジスタ + CF) → 汎用レジスタ
 39 0060 18C7               sbbb    %al,%bh
 40 0062 6619FB             sbbw    %di,%bx
 41 0065 19CF               sbbl    %ecx,%edi
 42                 # 汎用レジスタ - (メモリ + CF) → 汎用レジスタ
 43 0067 1A1D0000           sbbb    dtByte,%bl
 43      0000
 44 006d 661B3501           sbbw    dtWord,%si
 44      000000
 45 0074 1B0D0300           sbbl    dtDWord,%ecx
 45      0000
 46                 # メモリ - (汎用レジスタ + CF) → メモリ
 47 007a 18350000           sbbb    %dh,dtByte
 47      0000
 48 0080 66190D01           sbbw    %cx,dtWord
 48      000000
 49 0087 191D0300           sbbl    %ebx,dtDWord
 49      0000
 50          # アキュムレータ - (イミディエイト + CF) → アキュムレータ
 51 008d 1C12               sbbb    $0x12,%al
 52 008f 661D3412           sbbw    $0x1234,%ax
 53 0093 1D785634           sbbl    $0x12345678,%eax
 53      12
 54          # 汎用レジスタ - (イミディエイト + CF) → 汎用レジスタ
 55 0098 80DF12             sbbb    $0x12,%bh
 56 009b 6681D934           sbbw    $0x1234,%cx
 56      12
 57 00a0 81DF7856           sbbl    $0x12345678,%edi
 57      3412
 58                 # メモリ - (イミディエイト + CF) → メモリ
 59 00a6 801D0000           sbbb    $0x12,dtByte
 59      000012
 60 00ad 66811D01           sbbw    $0x1234,dtWord
 60      00000034
 60      12
 61 00b6 811D0300           sbbl    $0x12345678,dtDWord
 61      00007856
 61      3412
 62                 #*************************************
 63                 #       CMP
 64                 #*************************************
 65                 # 汎用レジスタ - 汎用レジスタ
 66 00c0 38C7               cmpb    %al,%bh
 67 00c2 6639FB             cmpw    %di,%bx
 68 00c5 39CF               cmpl    %ecx,%edi
 69                 # 汎用レジスタ - メモリ
 70 00c7 3A1D0000           cmpb    dtByte,%bl
 70      0000
 71 00cd 663B3501           cmpw    dtWord,%si
 71      000000
 72 00d4 3B0D0300           cmpl    dtDWord,%ecx
 72      0000
 73                 # メモリ - 汎用レジスタ
 74 00da 38350000           cmpb    %dh,dtByte
 74      0000
 75 00e0 66390D01           cmpw    %cx,dtWord
 75      000000
 76 00e7 391D0300           cmpl    %ebx,dtDWord
 76      0000
 77                 # アキュムレータ - イミディエイト
 78 00ed 3C12               cmpb    $0x12,%al
 79 00ef 663D3412           cmpw    $0x1234,%ax
 80 00f3 3D785634           cmpl    $0x12345678,%eax
 80      12
 81                 # 汎用レジスタ - イミディエイト
 82 00f8 80FF12             cmpb    $0x12,%bh
 83 00fb 6681F934           cmpw    $0x1234,%cx
 83      12
 84 0100 81FF7856           cmpl    $0x12345678,%edi
 84      3412
 85                 # メモリ - イミディエイト
 86 0106 803D0000           cmpb    $0x12,dtByte
 86      000012
 87 010d 66813D01           cmpw    $0x1234,dtWord
 87      00000034
 87      12
 88 0116 813D0300           cmpl    $0x12345678,dtDWord
 88      00007856
 88      3412
 89                 #*************************************
 90                 #       NEG
 91                 #*************************************
 92 0120 F6D9               negb    %cl
 93 0122 66F7D9             negw    %cx
 94 0125 F7D9               negl    %ecx
 95 0127 F61D0000           negb    dtByte
 95      0000
 96 012d 66F71D01           negw    dtWord
 96      000000
 97 0134 F71D0300           negl    dtDWord
 97      0000
 98                 #*************************************
 99                 #       DEC
100                 #*************************************
101 013a FEC9               decb    %cl
102 013c 6649               decw    %cx
103 013e 49                 decl    %ecx
104 013f FE0D0000           decb    dtByte
104      0000
105 0145 66FF0D01           decw    dtWord
105      000000
106 014c FF0D0300           decl    dtDWord
106      0000
```

ード同上をSBB命令で減算することで，64ビット長の減算が行えます．このように，SBB命令を必要な回数使うことで，ビット数が多い整数値の減算を行うことが可能となります．

● CMP命令

CMP命令は，オペランドで指定された二つの整数値を比較するための命令で，指定されたオペランドは，二つとも転送元となり変化しません．

CMP命令の動作は，転送元1をSOU1で，転送元2をSOU2で表すと，

SOU1 − SOU2

という減算のみを行い，減算結果は捨てられます．

アセンブラの記述上，転送元1(SOU1)をsou1，転送元2(SOU2)をsou2とした場合，MASMでは，

```
CMP    sou1, sou2
```

と記述し，gasではsou1とsou2が逆になり，

```
CMP    sou2, sou1
```

と記述します．

転送元1(SOU1)には汎用レジスタやメモリ上の値が指定できます．そして，転送元2(SOU2)には汎用レジスタやメモリ上の値，イミディエイトが指定できます．

ただし，メモリ上の値は，二つ指定できないため，一つのオペランドでメモリ上の値を指定した場合は，もう一方のオペランドは，汎用レジスタかイミディエイトとなります．

そのため，CMP命令のバリエーションは，

- 汎用レジスタ−汎用レジスタ
- 汎用レジスタ−メモリ
- メモリ−汎用レジスタ
- 汎用レジスタ−イミディエイト
- メモリ−イミディエイト

となります．

CMP命令実行後，ステータスフラグ(OF，SF，ZF，AF，PF，CF)は，減算結果にしたがい設定されます．ステータスフラグの設定のされ方は，SUB命令とまったく同じになっています．

はじめに述べたようにCMP命令は，指定された二つの整数値を比較するのに使用されます．そして，その結果としてステータスフラグに設定される，二つの整数値の大小関係を得ることが目的の命令です．

表2は，CMP命令で指定されたオペランドの二つの整数値の大小関係と，CMP命令実行後のステータスフラグの状態を示します．

実際のMASMでのCMP命令の記述例を**リスト3**，gasでのCMP命令の記述例を**リスト4**に示します．

〔表2〕CMP命令で比較される二つの値の大小関係とステータスフラグの状態

値の符号	値AとBをCMPで比較した結果，設定されるフラグ(CMPはA−Bと演算)	値AとBの大小関係	条件ジャンプなどのニモニックで使用される条件を表す文字
なし	ZF=1	A = B	E
	ZF=0	A ≠ B	NE
	CF=0 and ZF=0	A > B	A
	CF=0	A ≧ B	AE
	CF=1	A < B	B
	CF=1 or ZF=1	A ≦ B	BE
付き	ZF=1	A = B	E
	ZF=0	A ≠ B	NE
	ZF=0 and SF=OF	A > B	G
	SF=OF	A ≧ B	GE
	SF ≠ OF	A < B	L
	ZF=1 or SF ≠ OF	A ≦ B	LE

● NEG命令

NEG命令のオペランドは転送先一つで，オペランドで指定された転送先の汎用レジスタあるいはメモリ上の値を2の補数した値で置き換えます．

つまり，オペランドの転送元をDESTで表すと，

DEST ← − DEST

という演算を行います．

この−DESTと演算は，0−DESTの減算と同等の動作となります．

NEG命令を実行すると，CFは元のDESTがゼロならCF＝0に設定され，元のDESTがゼロ以外ならCF＝1に設定されます．CFを除くステータスフラグ(OF，SF，ZF，AF，PF)は，演算結果にしたがい設定されます．

実際のMASMでのNEG命令の記述例を**リスト3**，gasでのNEG命令の記述例を**リスト4**に示します．

● DEC命令

DEC命令もオペランドが転送先一つで，オペランドで指定された転送先の汎用レジスタあるいはメモリ上の値を−1する命令です．

つまり，オペランドの転送元をDESTで表すと

DEST ← DEST − 1

という減算を行います．

DEC命令を実行すると，CFを除くステータスフラグ(OF，SF，ZF，AF，PF)は，演算結果にしたがい設定されます．つまり，DEC命令ではCFは変化しません．そのため，DEC命令の実行により最上位ビットで桁借りが発生した場合でも，その桁借りはCFには設定されません．

実際のMASMでのDEC命令の記述例を**リスト3**，gasでのDEC命令の記述例を**リスト4**に示します．

*　　　　*

次回は，2進算術命令の乗除算命令と10進算術命令について説明する予定です．

おおぬき・ひろゆき　大貫ソフトウェア設計事務所

保護機能をもったμITRON仕様準拠カーネル

Hyper ITRONとμITRON4.0/PX仕様の解説

金田一勉

背景

身のまわりを見渡すと，じつに数多くの組み込みシステムが存在します．家庭内にある電気で動作してボタンや表示が付いた製品の多くにはマイコンが搭載され，組み込みシステムが動作しています．通勤通学の途中にも，会社や学校の中にもそういった機器が数多く見られます．また，携帯機器の中でも組み込みシステムが動作しています．それらのほとんどは，プログラムの大きさが数K～数百Kバイト程度であり，そのシステム全体をメーカーで把握することができます．

しかし，携帯電話を代表として，近年システムが巨大化していく傾向が見受けられます．これはいろいろな機能を追加し，使う側へより多くのサービスを提供しようとしているためです．こうなると，プログラムの大きさは1Mバイトを超え，数十Mバイトになる場合もあります．

ところで，組み込みシステムに各種の機能を実装するといった場合，マルチタスクOSの採用が一般的です．マルチタスクOSを利用することでシステムのプログラミングを簡略化することができます．

日本の組み込みシステムの多くはμITRON仕様準拠のOSを採用しています．これは，仕様がオープンなので理解が容易であること，数多くのメーカーから仕様に準拠したOSが開発されていて移行が容易であること，数多くのCPUに実装されていることなどが採用の理由だと思われます．

先の携帯電話のように，システムに組み入れる機能が増加してタスク数が100以上になると，すべてのプログラムを把握することが困難になってきます．また，ミドルウェアをほかから購入してシステムに組み込む場合もあります．

メーカーはシステムの試験を大量に行い，プログラムの不具合を見つけて修正します．しかし，システムが大きくなるほどプログラムの不具合を見つけることが難しくなります．そこで，万が一プログラムに不具合があっても，システム全体に及ぼす影響を最小限に食い止める策も考える必要があります．

そのために，保護機能を提供してその要望に応えようという動きがあります．保護機能を導入することでプログラムの不具合を早期に発見することができ，試験に要する期間を短くすることもできます．

本稿では，(株)エルミックシステムが開発・販売する，保護機能を搭載しμITRON3.0仕様に準拠した「Hyper ITRON」について解説します．

保護機能

「Hyper ITRON」は，エルミックシステムが独自に仕様を策定した，二つの保護機能をもっています．

- カーネルオブジェクト保護
- メモリ保護

μITRON仕様では，タスクやイベントフラグなどカーネルが管理する機能をカーネルオブジェクトと呼んでいます．プログラムが各種のカーネルオブジェクトに対して要求する際に，誤って別のカーネルオブジェクトに要求してしまうと，システムの動作に悪影響を及ぼすことがあります．そこで，これを防ぐためのしくみをカーネルオブジェクト保護といいます．一般に，このような誤りは初期の段階で発見することができます．

メモリ保護は，ソフトウェアがもつ機能だけでは実現できず，MMU(Memory Management Unit)を利用します．MMUに対しプログラムがアクセスできるメモリ領域の情報を設定し，プログラムが不正なメモリ領域をアクセスした場合にそれを検出し，警告を出したり不正な処理を行ったプログラムを停止させることが可能です．

ハードウェアへのアクセスとカーネル情報

- ハードウェアへのアクセス

μITRONのプログラムでは，ハードウェアアクセスは通常タスクと割り込み処理で行います．

「Hyper ITRON」では，ハードウェアへのアクセスは，入出力ドライバ形式を推奨しています．入出力ドライバとは，一般にデバイスドライバと呼ばれているものです．

入出力ドライバでは，ハードウェアへのアクセスを担当し，タスクからの要求を受けつけて処理を行います．入出力ドライバ

には割り込み処理も含んでいます．これは，タスクで行う処理とハードウェアへの制御処理とを分離する目的のために，カーネルで入出力ドライバを管理する機能をもっています．

● カーネル情報

タスクが容易にアクセスできるメモリ領域には，カーネルの管理情報を置かないようにしています．たとえば，メモリプールの空きメモリブロック管理やメイルボックスのメッセージ管理などです．タスクの誤動作によりカーネルの管理情報が破壊されると，カーネルが誤動作してシステム全体に影響が及んでしまいます．それを防ぐために，別に管理領域をもっています．

● システムコールのアドレスパラメータ

システムコールで指定されるアドレスパラメータの正当性をチェックします．ここでのアドレスパラメータとは，get_tid（タスクのID番号の取得）などでカーネルからの情報を格納するメモリ領域を指定するアドレスをいいます．カーネルは指定されたアドレスに情報を格納しますが，そのアドレスが不正な場合，カーネルを介して不正なメモリ領域を破壊することになります．

これを防ぐため，アドレスパラメータのチェックを行い，アクセスできないメモリ領域が指定された場合，エラーを返します．

● システムカーネルオブジェクト

「Hyper ITRON」では，負数のID番号の割り当てができるようにしており，システムカーネルオブジェクトとして扱います．負数のID番号を割り当てたシステムカーネルオブジェクトは，負数のID番号をもつタスクかタスク独立部（入出力ドライバなど）からのアクセスのみ許可されます．

システムカーネルオブジェクトには，正数のID番号を割り当てたタスクからのアクセスは禁止されます．

こうすることで，システムにとって重要なカーネルオブジェクトを保護することができます．

ソフトウェアグループ管理

保護機能を導入するために，保護の単位としてグループを設けます．

このグループには，タスクやイベントフラグなどのカーネルオブジェクトを所属させます．このグループ単位に管理する機能を「ソフトウェアグループ管理」といっています．

システムには，通信を行う機能や表示を行う機能，データを管理する機能などがあります．設計を行ううえでは，システムがもっている各種の機能単位でグループを形成するようにします．

このグループ単位でカーネルオブジェクト保護を行います．カーネルオブジェクト保護とは，システムコール時に指定するID番号に対応するカーネルオブジェクトへの不正なアクセスを防止するものです．

「ソフトウェアグループ管理」では，グループ単位にかたまりを形成します．このグループ単位にカーネルオブジェクト保護とソフトウェア部品の形成を推奨することができます．

● ID番号管理

「ソフトウェアグループ管理」では，従来のμITRON仕様でのID番号の管理をグローバルID番号といい，このグローバルID番号を割り当て，カーネルオブジェクトへのアクセスは自由になっています．グローバルID番号とは別に，グループごとにID番号をローカルに管理するローカルID番号があります．ローカルID番号に割り当てたカーネルオブジェクトは，ほかのグループからアクセスすることができません．

従来は，このようなソフトウェア部品を組み込む場合，その部品を構成するカーネルオブジェクトすべてにID番号を割り当てる必要がありました．「ソフトウェアグループ管理」を導入することで，グローバルID番号を割り当てることが少なくなり，割り当てる際の手間が少なくなります．

● カーネルオブジェクト保護

「ソフトウェアグループ管理」において，グループごとにグループID番号を設定するようになっています．

グループID番号の0番（グループ0）は，従来のID番号管理と互換の管理を行います（グローバルID番号管理）．このグループ0に所属するカーネルオブジェクトは，ほかのグループから自由にアクセスができます．

グループID番号1番以降のものは，それぞれのグループでID番号をローカルに管理することができます（ローカルID番号管理）．

グループ間で情報を交換する場合は，グループ0のカーネルオブジェクトを介すようにします．

グループに所属するタスクにグループ0のID番号を付けると，そのタスクは，グループ0のカーネルオブジェクトと，そのグループに所属するカーネルオブジェクトの両方にアクセスできるようになります．

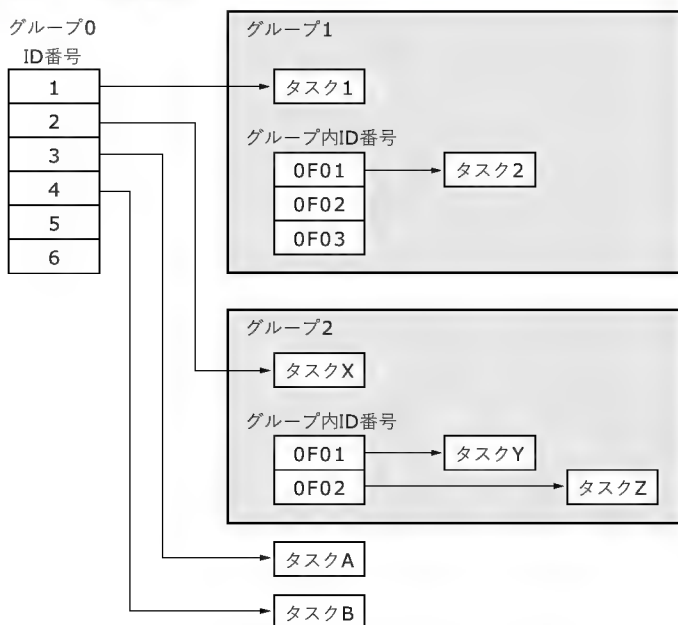
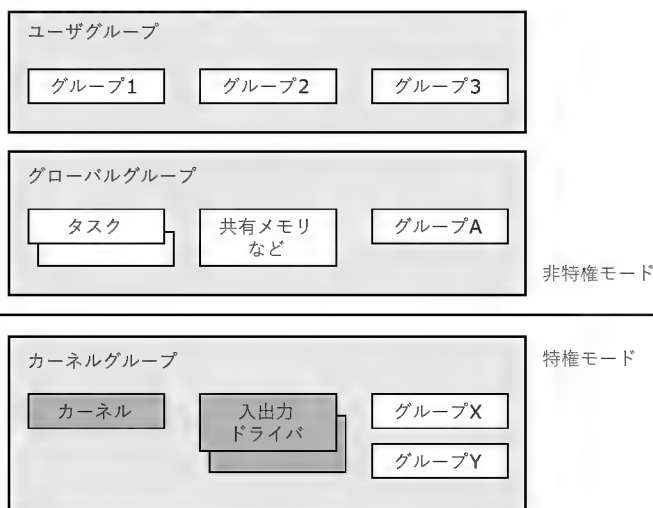
グループ0のID番号をもつタスクは，グループ0のカーネルオブジェクトにはアクセスできますが，ほかのグループのローカル管理されているカーネルオブジェクトの指定はできません．

グループ内にローカル管理されているID番号をもつタスクは，グループ0のカーネルオブジェクトにはアクセスできますが，ほかのグループに所属するカーネルオブジェクトの指定はできません．

ソフトウェアグループでのID番号管理の例を**図1**に示します．グループ内ID番号において，0FxxHの0FHは，グループ内管理を示すマークとします．

グループ1に所属するタスク1は，タスクAやタスク2に対して要求を行うことができますが，グループ2にあるタスクYには要求することはできません．また，グループ1に所属するタスク2は，グループ2に所属するタスクYに要求することはできません．グループ0に所属するタスクAは，タスク1やタスクXに要求できますが，タスク2やタスクYに要求することはできません．

図1で，グループ1にタスク3が追加されたとしても，グルー

〔図1〕ID番号管理

〔図2〕メモリ保護グループ

プ1内のローカルID番号(0F02)に割り当てを行うだけなので，他グループのID番号の割り当てへの影響はありません．

● ソフトウェア部品

「ソフトウェアグループ管理」によりID番号をそのグループ内で管理することにより，グループ単位に扱うことができます．システムにそのグループを組み込む際に，ほかのグループとの情報を交換するカーネルオブジェクトに対し，グローバルID番号を割り当てます．グループ内だけで扱うカーネルオブジェクトには，ローカルID番号を割り当てます．

カーネルオブジェクトをグループ単位でローカルに管理することで，その部品が機能アップなどによりカーネルオブジェクトの構成が変更になっても，ローカルID番号に割り当てを行うだけで，グローバルID番号への割り当てが不要になります．このことは，グループ単位のカーネルオブジェクトの構成を隠蔽できることを示します．

従来であれば，メーカーが部品を提供し，それをシステムに組み込んでもらう場合，その構成をすべて公開し，ID番号をユーザーに割り当ててもらいます．部品の機能アップなどにより構成が変わると，ユーザーに対して再割り当てをしてもらいます．

「ソフトウェアグループ管理」を導入することで，メーカーはカーネルオブジェクトの構成をすべて公開する必要がなくなり，また機能アップなどによる構成の変更があってもユーザーへの影響が少なくなります．

ローカルID番号は部品ごとに固定値になるので，その部品をバイナリで供給することができるようになります．従来であれば，システムに部品を組み込む場合，部品を構成するカーネルオブジェクトのID番号が組み込むシステムにより変わるので，ソースを公開し，ID番号を決定してもらってコンパイルし直す必要があります．部品のソースを公開することができない場合でも，バイナリでの供給が可能であれば，ユーザーへの提供が可能になります．

また，ローカルID番号を割り当てたカーネルオブジェクトはほかのグループからのアクセスができなくなるので，不正なアクセスによる誤動作を防ぐことができ，万が一誤動作してもその原因を追求することが容易になります．

メモリ保護

メモリ保護は，MMUの保護機能を利用します．仮想アドレスと物理アドレスとは1対1の対応としているので，MMUがもつアドレス変換機能は利用しません．

MMUを利用するため，MMUのページ単位でメモリ保護を行います．そのため，保護の単位はMMUのページ単位に配置します．

● 保護グループ

メモリ保護でも，属性によりグループ分けを行っています．グループには**図2**のようにユーザグループ，グローバルグループ，カーネルグループがあります．

ユーザグループは，MMUの非特権モードで動作するソフトウェアグループのグループが所属します．このグループ単位で保護が行われ，ソフトウェアグループ間でのメモリ領域へのアクセスはできません．

ユーザグループに所属するグループは，そのグループが占めるメモリ領域とグローバルグループのメモリ領域に対してアクセスが可能です．ほかのグループのメモリ領域，カーネルグループのメモリ領域，システムで定義したメモリ領域以外にアクセスした場合，メモリ保護違反として扱われます．

グローバルグループは，すべてのグループからアクセスができる領域で，非特権モードで動作します．グループ間で情報交換

〔表1〕アクセス許可/禁止のメモリ領域

	アクセスが許可されるメモリ領域	アクセスが禁止されるメモリ領域
ユーザグループに所属するグループ	●そのグループが占めるメモリ領域 ●グローバルグループが占めるメモリ領域	●他のユーザグループが占めるメモリ領域 ●カーネルグループが占めるメモリ領域 ●システムで定義した以外のメモリ領域
グローバルグループに所属するグループ	●グローバルグループが占めるメモリ領域	●ユーザグループが占めるメモリ領域 ●カーネルグループが占めるメモリ領域 ●システムで定義した以外のメモリ領域
カーネルグループに所属する入出力ドライバやソフトウェアグループ	●ユーザグループが占めるメモリ領域 ●グローバルグループが占めるメモリ領域 ●カーネルグループが占めるメモリ領域	●システムで定義した以外のメモリ領域

するためのメモリ領域や共有で使用する関数など(通常のライブラリ関数も含む)を所属させます．グローバルグループにもソフトウェアグループのグループを所属させることができますが，アクセスできるメモリ領域はグローバルグループが占めるメモリ領域だけになります．

カーネルグループは，MMUの特権モードで動作し，カーネル本体や入出力ドライバが所属します．カーネルグループは特権モードで動作するので，基本的にメモリ領域へのアクセスは自由になります．

Hyper ITRONでは，ハードウェアへのアクセスは，入出力ドライバで行うように推奨しています．一般のタスクはユーザグループに所属するため，ハードウェアへのアクセスを行うことができません．入出力ドライバは，タスクからの要求を受け付けて入出力処理を行います．入出力ドライバはカーネルグループに所属しているので，ハードウェアへのアクセスを行うことができます．

カーネルグループにソフトウェアグループのグループを所属させることもできます．タスクでハードウェアへのアクセスを行いたい場合は，カーネルグループに所属させます．

各グループとアクセスが許可されるメモリ領域と禁止されるメモリ領域を**表1**に示します．

MMUの保護機能を利用することで，コード領域への書き込みアクセスを禁止することができます．アクセスが禁止されているメモリ領域にアクセスした場合やコード領域への書き込みアクセスが行われた場合は，メモリ保護違反処理を行います．ユーザグループやグローバルグループに所属するタスクがメモリ保護違反を行った場合には，警告を出力して対象のタスクを停止します．そして，カーネルグループに所属する入出力ドライバやタスクがメモリ保護違反を行った場合には，警告を出力してシステムを停止します．

MMUの制御はMMUドライバが行っており，カーネル処理と分離しています．システムのデバッグ時はメモリ保護機能を使ってプログラムを動作させ，デバッグ終了後にMMUドライバの登録をやめて，メモリ保護機能なしで動作させるということができます．このことで，MMUからのTLB(Translation Lookaside Buffer)ミスによる例外処理がなくなるため，その分オーバヘッドがなくなり，システムの動作スピードは向上します．

● セクション

メモリ保護の対象になるメモリ領域をセクションに分離します．ここでは，日立製作所製SH-3/4でのセクションの分離について説明します．SH-3/4では，日立製作所製の開発環境を利用します．

各グループごとにセクション名を指定してコンパイルし，そのセクションの配置を指定してリンクします．

グループ0のセクション名は，P，C，D，R，Bとしています(デフォルトのセクション名)．

グループ1以降のセクション名は，GRPx_P，GRPx_C，GRPx_D，GRPx_DRAM，GRPx_Bとします(xはグループのID番号に対応)．

カーネルグループのセクション名は，KRN_P，KRN_C，KRN_D，KRN_DRAM，KRN_Bとします．

メモリ領域の属性によるセクションの名前の付け方を示します．

- PもしくはXXX_P　：プログラムセクション(RO)
- CもしくはXXX_C　：コンスタントセクション(RO)
- DもしくはXXX_D　：ROM領域に格納するデータセクション(RO)
- RもしくはXXX_DRAM　：データセクションをコピーするセクション(RW)
- BもしくはXXX_B　：初期値なしセクション(RW)

グループをカーネルグループに所属させる場合は，セクション名をKRN_xxxとします．

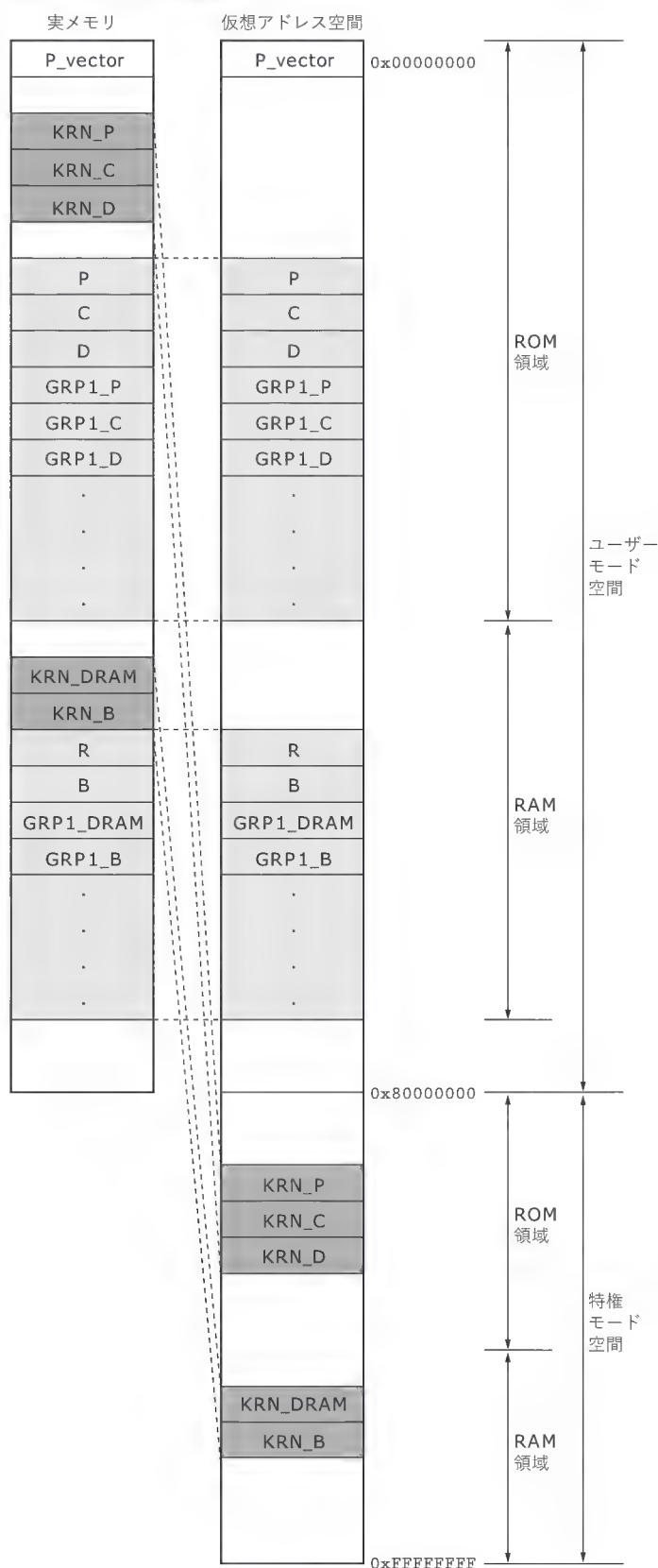
各セクションは，MMUのページ単位である4096バイト単位にアラインするようにしています．特定のアドレスに配置する場合を除き，リンク時にセクションの登録を行うことで，自動的に4096バイト単位に配置されるようになっています．

セクションの配置例を**図3**に示します．

● メモリ保護違反

メモリ保護違反処理は，MMUドライバで行います．MMUドライバで行うメモリ保護違反処理は，ユーザーで記述することができます．出荷時のMMUドライバでは，違反の要因をメッセージとして出力し，対象のタスクを停止させます．

タスクでメモリ保護違反が発生した場合，メモリ保護違反処理で出力するメッセージの例を**図4**に示します．

〔図3〕セクション配置例

〔図4〕メモリ保護違反メッセージ例

① 不正アクセスの要因メッセージ
② 不正アクセスを行ったプログラムカウンタ(PC)
③ 不正アクセスの対象アドレス
④ 不正アクセスを行ったタスクのID番号(グループID番号)

不正アクセスの要因により，以下のようなメッセージを出力するようにしています．

- TLB Miss (Read)　：TLB 無効例外(読み出し)
- TLB Miss (Write)　：TLB 無効例外(書き込み)
- Read Access Violation　：TLB 保護違反(読み出し)
- Write Access Violation：TLB 保護違反(書き込み)
- Address Error (Read)　：アドレスエラー(読み出し)
- Address Error (Write)：アドレスエラー(書き込み)

上記のメッセージはシリアルに接続する端末に出力しますが，ログを残すなどといったシステム固有の処理をユーザーで記述することができます．

- 利用例

カーネルオブジェクト保護とメモリ保護を使用した例を示します．

1）ソフトウェア部品化

ソフトウェア部品として，グループ単位に各種のシステムで利用することができます(**図5**)．

システムAでは，グループ3として登録していたグループをシステムBではグループZとして登録することができます(ソフトウェアグループのID番号管理による部品化の推奨)．

2）オプション機能の追加

システムにオプション機能を追加しても，ID番号の割り当ては最小限で済みます．

また，追加するオプションの構造のすべてを公開しなくても，システムへの組み込みができます(**図6**)．

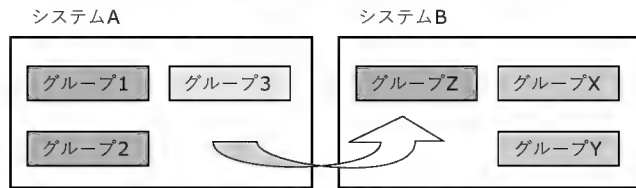
3）プログラムの共有

複数のチャネルで同じプロトコル通信を行うようなプログラムの場合，同じプログラムを実行させ，その動作環境を変えることができます(**図7**)．

これは，グループ内のカーネルオブジェクトは，ローカルID番号管理により別々のカーネルオブジェクトを割り当てることができることと，ローカルID番号は固定値にすることができることを利用したものです．**図7**の点線のタスクは同じプログラムを指定することができます．

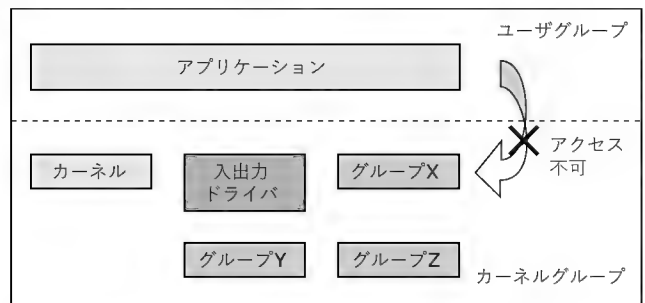
4）デバッグ支援

グループの試験中はユーザグループに所属させ，試験後はカーネルグループに移行します(**図8**，カーネルグループに移行さ

〔図5〕ソフトウェア部品化の例

〔図6〕オプション機能追加の例

〔図8〕デバッグ支援の例

〔図7〕プログラムの共有の例

〔図9〕基幹システムの保護の例

〔図10〕非特権モードで動作するタスクと特権モードで動作するタスクを混在させる例

せることにより，TLBミスによるオーバヘッドを削減）.

5）基幹システムの保護

アプリケーションから，基幹システムのメモリ領域やカーネルオブジェクトを保護することができます（**図9**）.

システムにとって大事なデータをアプリケーションの誤動作から保護することができます.

● 応用

これまで，メモリ保護の単位は，ソフトウェアグループで定義するグループ単位として説明してきました．ソフトウェアグループのグループは，ID番号の管理のためのまとまりです．メモリ保護は，セクションとそのグループを管理することで実現しています．一つのグループに非特権モードで動作するタスクと特権モードで動作するタスクとを混在させることができます.

特権モードで動作させるタスクのコンパイル時とシステムに登録する際の設定を行うことで混在させることができます．ただし，非特権モードで動作するタスクは，特権モードで動作するタスクのメモリ領域に対してアクセスすることはできません．情報の交換は，必ず特権モードで動作するタスクが非特権モードのメモリ領域を参照するようにします.

このことで，非特権モードで動作するタスクと，特権モードで動作しハードウェアに直接アクセスするタスクとを一つのグループに混在させることができます（**図10**）.

μITRON4.0/PX仕様

トロン協会では，2001年1月からメモリ保護に関する仕様の策定を開始し，2002年6月にその仕様を公開しています.

保護を行う際の単位を「保護ドメイン」としています．保護ドメインには，カーネルオブジェクトを含めます．保護は，保護ドメイン単位でのアクセスの許可/不許可の指定になっています．この指定は，アクセス方法により「書き込み系アクセス/読み出し系アクセス/参照系アクセス/管理系アクセス」に分かれます．それぞれのアクセス系の許可/不許可のパターンを形成し，その

〔表2〕保護仕様の比較

	μ ITRON4.0/PX	Hyper ITRON
保護単位	保護ドメイン カーネルコンフィギュレーションファイルでの定義がスタンダード	ソフトウェアグループ カーネルオブジェクトを登録する際にグループを指定
保護単位とカーネルオブジェクトの定義	スタンダードではCFGファイルに記述	テーブルに情報を記述しシステムに登録
カーネルオブジェクト保護	カーネルオブジェクトの登録時に指定するアクセス許可ベクタによるアクセス許可／不許可の設定	「ソフトウェアグループ管理」によりローカルに管理されているID番号への割り当て
メモリ保護	MMU/MPUなど，メモリアクセスを監視するハードウェアで保護を実現	MMUを利用して保護を実現
ハードウェアへのアクセス	拡張SVCでのサポート	入出力ドライバでのサポート
特徴1	保護メモリプール，保護メイルボックスによる保護されたメモリ領域による情報交換機能	「ソフトウェアグループ管理」によるソフトウェアの部品化への推奨
特徴2	カーネルオブジェクト個々に対し，アクセス許可/不許可が細かく設定可能	カーネル処理とMMU制御処理が分離しているので，メモリ保護機能の取り外しが可能
特徴3	メモリ領域に対してアクセス許可ベクタを設定できる	負数のID番号のカーネルオブジェクトは負数のID番号をもつタスクかタスク独立部からのアクセスのみ許可
システムコールのアドレスパラメータのチェック	あり	あり (ユーザーの指定により省略可能)
カーネル情報の保護	あり	あり

パターンをアクセス許可パターンといいます．アクセス系によりそれぞれのパターンを決定し，それをまとめてアクセス許可ベクタと呼んでいます．

保護ドメインには，ユーザドメイン，システムドメイン，カーネルドメインおよびドメイン無所属があり，保護のレベルが異なります．

本稿では，タスクからカーネルに対する要求を「システムコール」という名称で統一しています(μITRON4.0仕様では「サービスコール」という名称になっている)．

● カーネルオブジェクト保護

μITRON4.0/PX仕様でのID番号の管理は従来と同じです．

μITRON4.0/PX仕様のスタンダードはシステムコンフィギュレーションファイルでシステムの構造を記述します．このシステムコンフィギュレーションファイル(以下CFGファイル)に，保護ドメインの囲みにしたがってカーネルオブジェクトを定義していきます．

保護ドメイン内に登録するカーネルオブジェクトは，デフォルトではその保護ドメインに所属するタスクからしかアクセスできません．ほかの保護ドメインからのアクセスを許可するカーネルオブジェクトには，アクセス許可ベクタを定義します．アクセス許可ベクタを定義する際に，アクセスを許可する保護ドメインを指定します．

保護ドメインの囲み以外のところでカーネルオブジェクトを定義すると，それはドメイン無所属となり，基本的にどのタスクからの要求も許可されます．ドメイン無所属には，タスクを含めることはできません．

このように，カーネルオブジェクト保護は，カーネルオブジェクト個々に対してアクセス許可ベクタを設定し，システムコール時にアクセスの許可/不許可をチェックし，不正なアクセスを防ぐ方法です．

● メモリ保護

μITRON4.0/PX仕様では，保護ドメインを保護の単位としています．

メモリ保護機能の実現では，ハードウェアのメモリ保護機能を利用します．一般によく知られるMMU(Memory Management Unit)には，アドレス変換機能もありますが，仕様ではその機能は利用しておらず，仮想アドレスと物理アドレスは1対1の対応になっています．

CFGファイルに記述した保護ドメインは，コンフィギュレータによりメモリ保護ユニットのアドレス単位に合わせて配置されます．

メモリ領域に対してもアクセス許可ベクタを設定します．保護ドメインが占めるメモリ領域は，デフォルトでその保護ドメイン内のタスクからのみアクセスが許可され，ほかの保護ドメインからのアクセスは許可されません．

保護ドメインが占めるメモリ領域以外のメモリ領域に対し，アクセス許可ベクタを設定することで，アクセスの許可/不許可を指定します．これらのメモリ領域のアクセス許可ベクタは，システムコール時に指定するアドレスパラメータのチェックやTLBミス時の処理に参照されます．

● 追加機能

μITRON4.0/PX仕様では，大容量のデータを保護ドメイン間で保護された状態で送受信できる機能を定義しています．それが保護メモリプールと保護メイルボックスです．これは，ページ単位でメモリを管理するMMUを使用する場合に利用できます．

保護メモリプールは，MMUのページ単位に管理された可変長のメモリプールになります．タスクが保護メモリプールから保護メモリブロックを取得した場合，取得したタスクが所属する

保護ドメインに所属するタスクからのアクセスが許可されます．取得した保護メモリブロックは，ほかの保護ドメインからのアクセスは禁止されます．

保護メモリブロックにデータを格納し，保護メイルボックスに送信します．保護メイルボックスは，保護メモリブロックのアドレスのみ送信することができます．タスクが保護メモリブロックを保護メイルボックスに送信すると，保護メモリブロックのメモリ領域へのアクセスが許可されなくなります．保護メイルボックスから保護メモリブロックを受信すると，受信したタスクが所属する保護ドメインに所属するタスクからメモリ領域へのアクセスが許可されます．保護メモリブロックを保護メモリプールに返却すると，返却した保護メモリブロックのメモリ領域へのアクセスは許可されなくなります．

これらの機能を利用することで，ほかのプログラムの誤動作による破壊から防ぐことができるメモリ領域を確保できることと，ドメイン間で保護された情報を渡すことができるようになります．

最後に，今回解説したHyper ITRONとμITRON/PXの比較を**表2**に示します．

今後の展開

「Hyper ITRON」は，カーネルオブジェクト保護とメモリ保護をもったμITRON仕様準拠のカーネルです．

ただ，メモリ保護を利用するには保護単位にメモリ領域を分ける必要があるため，ソースファイルに対して修正を加えていく必要があります．そのため，既存システムへの導入にはかなり敷居が高いものになっています．

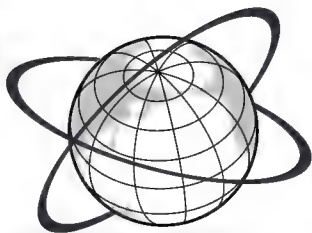
「ソフトウェアグループ管理」により，既存のAPIの変更なしでソフトウェア部品化を容易にする機能への要望が高くなっています．そのため，エルミックシステムでは，既存のμITRON仕様準拠のカーネルである「ELX-ITRON」に対し，「ソフトウェアグループ管理」を追加した「ELX-ITRON Neo」を開発中です．

参考文献

1）『μITRON4.0仕様　保護機能拡張（μITRON4.0/PX仕様）』，（社）トロン協会
2）「メモリ管理のしくみとプロセッサへの実装」，『デザインウェーブマガジン』，2002年9月号
3）『SH7709Aハードウェアマニュアル』，（株）日立製作所

■ **エルミックシステムのμITRON仕様準拠のカーネル製品についての照会と問い合わせ先**
URL：http://www.elmic.co.jp/
E-mail：info@elmic.co.jp

きんだいち・つとむ　（株）エルミックシステム

インターネットを使った音声通話の実現

IP電話システムの概要と現状

太田博之

加入電話はもう要らない?

「電話料金を月額固定に……」という見出しが，「IP電話」というキーワードとともに散見されるようになりました．では，そのIP電話というのは何なのでしょうか．

「050からはじまる電話番号でつながるらしい」，「インターネットでつなげるらしい」，「どうやら，通常電話からの着信ができないらしい」……という断片的な情報だけが行き交っています．このように正体不明？のIP電話ですが，すでにフュージョン・コミュニケーションズ(株)は，IP電話のテクノロジを使って，全国一律3分20円の電話サービスを提供しています．企業ユーザー向けには，沖電気工業(株)をはじめ大手電機メーカーがIPソリューションを提供しており，その目玉がIP電話になっています．

このような現在注目されているIP電話について解説します．

IP電話とは何か

IP電話とは，Internet Protocol(IP)の規格に準拠した形で，電話相当のサービスを提供する技術です．

IP電話では，以下の二つの技術が必要になります．

- 通話の相手に対しての接続，切断，通信管理を行うしくみ
- 音声をIPパケットに変換して相手に伝えるしくみ

この二つについて簡単に説明します．

通話の相手に対しての接続，切断，通信管理を行うしくみ(プロトコル)は，以下のとおりです．

- ダイヤルされた番号により相手を特定する
- 特定した相手に接続の意思を伝え，許可を受ける
- 通話中の課金管理
- 通話の終了

このようなプロトコルはIP電話にかぎらず，携帯電話，一般電話でも必要になります．IP電話ではこれらのプロトコルをIPの規格に沿った形で実現します(詳細は後述)．

もう一つの，音声をIPパケットに変換して相手に伝えるしくみを簡単に表すと**図1**のようになります．IP電話を使用するユーザー間では，授受される音声をディジタル化し，IPパケットに乗せて電話相手に送ります．相手側では，IPパケットのデータを再び音声に戻します．この流れを双方向に行い，電話の機能を実現します．

これまでの説明でわかるように，IP電話ではインターネットのリソース(設備)を使用するために，NTT，日本テレコム，KDDIなどが提供する公衆回線網も公衆回線交換機も必要としません．そのため，これまで発生していた，通話時間と相手先までの距離に依存する電話料金(課金)は発生しません．そのかわり，以下のような費用が発生します．

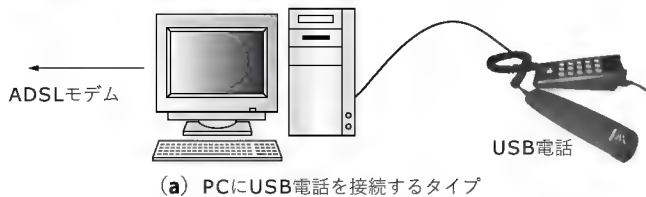
- インターネットを使用するためISP(Internet Service Provider)に支払う料金
- ADSLなどのインターネット接続のための回線使用にかかる費用
- 上記のプロトコルと音声伝達のしくみを実現するソフトウェア，ハードウェアにかかる費用

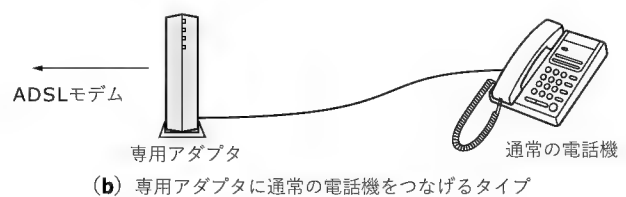
ただし現状では，これらの費用はいずれも月額固定で，通常の電話のような通話時間，相手先までの距離依存の電話料金は発生しません．IP電話は，料金の面で見ると，月額固定料金の使い放題の電話というイメージになります．

また，**図1**からもわかるとおり，IP電話のサービスを使用するためには，電話をかける側と電話を受ける側はインターネット網に接続していなければなりません，そのうえ，IP電話のための設備も必要になります．

まとめると，IP電話は通常の電話に比べて，月額固定料金の

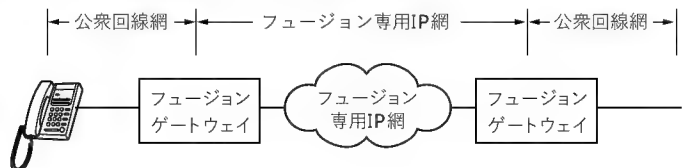
〔図1〕IP電話のしくみ

〔図2〕ブロードバンド型IP電話サービス

(a) PCにUSB電話を接続するタイプ

ADSLモデム
専用アダプタ
通常の電話機

(b) 専用アダプタに通常の電話機をつなげるタイプ

〔表1〕各CODECの特徴

CODEC	方式名称	ビットレート(bps)
G.711	PCM	48K, 56K, 64K
G726	ADPCM	16K, 24K, 32K, 40K
G729	CS-ACELP	8K
G.723.1	ACELP/MP-MLQ	5.3K, 6.3K

〔図3〕中継型IP電話(フュージョンの例)

使い放題の電話という大きなメリットがありますが，お互いにインターネットに接続され，IP電話のしくみをもった二者間でのみ通話可能という制限があります.

使用される技術

上記でも説明しましたが，IP電話を実現するためにはIP電話特有のプロトコルと，音声データとIPパケットを相互変換するしくみが必要になります.

● プロトコル

IP電話に使用されるプロトコルはいくつかの種類がありますが，H.323とSIPの2種類がほとんどです.

H.323は，音声，ビデオ，データなどのマルチメディア通信を行うための規格で，ITU-T〔各国の通信関連の省庁(日本の場合は総務省)の代表で構成される規格委員会〕で標準勧告されています.

SIPは1999年3月にIETF(Internet Engineering Task Force)より発表された規格で，H.323より後発のため，まだあまり普及は進んでいません．しかしながら，後発ゆえに転送機能や発信者番号通知機能など，H.323に比べて公衆電話網に近い機能を備えています．Windows XPに搭載されているWindows Messengerには，このプロトコルが使用されています.

現在の市場においてはH.323搭載機器が多いのですが，Windows XPに搭載が決まったことなどの状況から，SIP搭載機器は今後増えていくことが予想されます.

H.323は従来の電話網のプロトコルから発展してIPに対応させたのに対して，SIPはインターネット上の通信制御から発展し，電話としてのプロトコルを備えるようになりました．それぞれ，機能はほぼ同じなのですが，出自が違うということで両者が並び立っているのです.

● 音声のデータ化(CODEC)

音声のCODECはG.711/G.723.1/G.726/G.729が使われます．G.711は必須CODECで，その他の方式はオプションとなっています．それぞれのCODECの特徴を**表1**にまとめました．さまざまな種類がありますが，ビットレートが低いほど劣悪な環境でも安定した通話が可能になります．このため多くのIP電話ではG.723.1，G.729などがよく使われています.

どのCODECを使うかですが，電話をかけて通話が始まる前にプロトコルの中で照合を行い，自分と相手が共通にもっているCODECを選択するという手順になっています.

● IP電話サービスの種類

2002年9月末でブロードバンドユーザーが国内で約600万になり，今後も増加する傾向にあります．また，一説によるとブロードバンドユーザーの10%～30%がIP電話を使い始めているという統計もあります．ここでいっているIP電話もいろいろな形があります．現在発表されている，またはサービスを始めているものを紹介します.

IP電話の形としては，PCに専用端末(USB電話)をつないで使うタイプと専用アダプタをつないで通常の電話機を使用するタイプがあります(**図2**)．この二つは家庭内のADSLモデムに接続する方式で，ブロードバンド型と呼ばれています.

もう一つ，ブロードバンド型とは別に中継型と呼ばれるサービスがあります．中継型IP電話は，2001年の春からフュージョン・コミュニケーションズがサービスを開始しており，約200万ユーザーが使用しています．中継型IP電話は，フュージョンのゲートウェイまではNTTの公衆回線網を使用します．ゲートウェイでは音声をディジタル化し，IPパケットとしてフュージョン専用IP網を通じて相手のゲートウェイ→公衆回線網→電話端末に音声を届けます．このしくみを**図3**に示します.

また，接続相手についても同一ISPのユーザー同士のみ接続可能なタイプと，一般電話機に電話をかけることができるタイプの2種類があります(**表2**).

海外のIP電話事情

海外でのIP電話の先進国は，アメリカと韓国です．IP電話は

〔表2〕ISP別のサービス内容

サービス名/製品名	提供会社	使用方法	接続相手
KDDIインターネットIPフォンサービス	KDDI	専用アダプタが必要	一般回線(8.5円/3分)
FUSION IP-Phone for Biglobe	NEC	専用アダプタが必要	一般回線(8円/3分)
Yahoo BB Phone	Yahoo	専用アダプタが必要	一般回線(7.5円/3分)
Go2call	Nifty	PCに専用端末を接続して使う	一般回線(15円/3分)
Betarena	Nifty	PCにヘッドセットを接続して使用	Niftyユーザー同士(無料)
everyPhone	マネジメントジャパン	PCに専用端末を接続して使う(**写真1**)	ユーザー同士(無料) 一般回線も計画中

〔写真1〕everyPhoneで使用する専用端末

〔写真2〕ヘッドセット

既存の電話よりも安価なことが特徴ですが，アメリカと韓国の従来の電話会社は，これに対抗して国内電話を非常に安価に提供しています．

しかし，海外通話の場合は低料金でのサービスが困難なので，IP電話を使用する場合が多いようです．

韓国の場合，2001年まではPCとヘッドセット(**写真2**)を使用するユーザーが多かったのですが，今年に入ってUSB電話をPCに接続して使用するユーザーが増えてきています．ヘッドセットに比べてUSB電話のほうが格段に音質が良いといわれており，すでに台湾，韓国合わせて20社程度のUSB電話のメーカーが参入しています．

将来の展望

総務省は，IP電話に対して050で始まる番号を付与することを決め，2002年の9月から番号の申請受け付けを開始しました．本誌が出る頃には050の番号によるサービスが開始されていることでしょう．

このように050の番号を総務省はIP電話に付与するのですが，一般電話からIP電話への050番号による着信は，2003年以降になるようです．これはNTTの交換機に050の番号を登録して，ユーザーから050で電話がかかってきた場合に，対応するIP電話サービス会社に接続する必要があるからです．この動作は交換機のプログラムにより行われるのですが，対応に1年近くの時間がかかるようです．また，変更にかかる費用をどこが出すかについても検討する必要があるようです．

交換機のプログラムの変更がすべて終われば，通常の電話とほぼ同じ使い勝手になります．その時期は2003年秋以降といわれています．

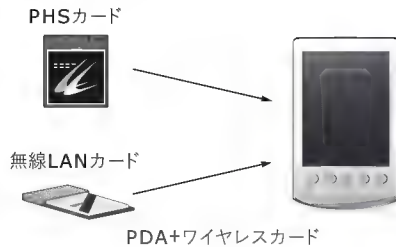
また，IP電話の将来としては以下のようなものが考えられます．

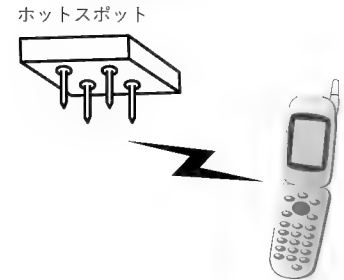
IP電話のサービスの中でPCにUSB電話をつないで使用するタイプがありました．この場合，IP電話のプロトコルと音声符号化はPC内のソフトウェア(これをクライアントソフトウェアと呼ぶ)で行います．PCに搭載する代わりに**図4**のようにPDAにクライアントをインストールし，PHSカードまたは無線LANカードをPDAに挿せば，PHSのサービスエリア，または無線LANのホットスポットでIP電話のサービスを受けることができます．これもモバイルIP電話の一つの形といえるでしょう．

また，PDAにBluetoothがついていて，USB電話の代わりにBluetooth電話端末を使用すれば，携帯電話型のモバイルIP電話になります．このときPDAは上着の内ポケットや鞄の中に入れておくことになるでしょう(**図5**)．

さらにクライアントに相当する部分をすべて携帯電話型端末に内蔵すれば，無線LANホットスポットで，PCやPDAを使わずにIP電話を使うことができます(**図6**)．

また，クライアントをゲームマシンにインストールしてUSB電話を接続すれば，IP電話として使用することも可能です(**図7**)．ゲームマシンでネットワーク型ゲームを行っている場合，ゲーム画面を凝視してゲームするだけではなく，電話で話しながら次の手を考えるなどの使い方をすると，さらにゲームが面白くなるでしょう．

〔図4〕PDA＋ワイヤレスカード

〔図5〕PDA＋Bluetooth

〔図6〕クライアントをすべて携帯電話端末に内蔵する

IP電話の例 ―― everyPhone

everyPhoneはマネージメント・ジャパン(株)が開発したオリジナルのVoIPシステムです．

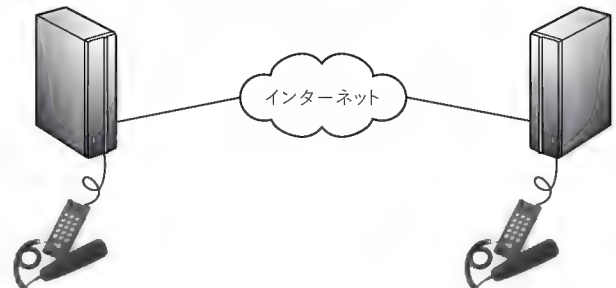
everyPhoneのシステムはクライアントソフトウェア，USB電話，ディレクトリサーバ，留守番電話サーバから構成されます．また，法人向けサービスのために代表電話機能，ADSLダイナミックIPアドレス管理，NAT機能をもっているPBX for everyPhoneを用意しています．クライアントはH.323に準拠しており，現在SIP対応版も開発中です．

everyPhoneはブロードバンド型のIP電話サービスで，以下のような特徴があります．

1) インターネット上にディレクトリサーバを置いているためにISP非依存のサービスが可能
2) ファイアウォールなどを備える法人ユーザーのためにクライアントがUPnP対応になっている
3) 同じく，法人ユーザーのために廉価なPBX機能をもった簡易サーバ(PBX for everyPhone)を提供している
4) クライアントはアップグレードに対応できるようにWebページからダウンロード可能
5) PCやクライアントが作動していない場合は，インターネット上の留守番電話サーバにメッセージが蓄積される(携帯電話の留守電サービスとほぼ同等のサービスを提供)

everyPhoneを使用すれば，PCとUSB電話とインターネット接続環境さえあれば，世界中のどこでも同じように電話をすることができます．

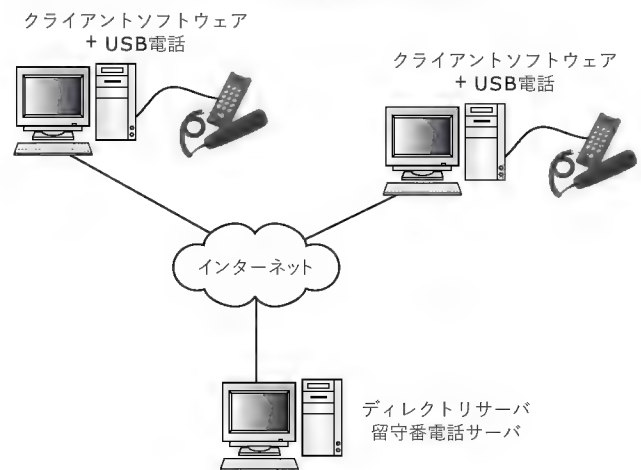
これを実現するため，everyPhoneはインターネット上にディレクトリサーバを設置しています(**図8**)．電話をかける場合，クライアントは相手がどこにいるかをディレクトリサーバに問い合わせます．ディレクトリサーバにはログインしたときのユーザーのIPアドレスが記録されているので，クライアントからの問い合わせに対して相手のIPアドレスを知らせます．

〔図7〕ゲームマシンを使用したIP電話

〔図8〕ディレクトリサーバを設置する必要がある

クライアントソフトウェア
＋USB電話

クライアントソフトウェア
＋USB電話

インターネット

ディレクトリサーバ
留守番電話サーバ

このように，相手がどこ(どのIPアドレス)にいるかを知ることができるので，自分や相手がどこにいても，いつもとかわらずに電話をすることができます．IP電話を使用すれば，世界中のすべての場所を自分のオフィスにすることが可能です．

おおた・ひろゆき　加賀電子(株)

C言語のソースコードを読みやすく整形する
― GNU indent

水野貴明

今回は，GNU indentというC言語用のソースコードフォーマッタを紹介する．これはソースコードをチェックして，改行やインデントを一定の規則にもとづいてやり直してくれるものだ．これを使うことで，自分とは異なるスタイルで書かれたソースコードを読みやすく整形することができる．

DATA

名称：GNU indent

Webサイト：http://home.hccnet.nl/d.ingamells/beautify.html

現在のバージョン：2.28a

GNU indentとは

GNU indentは，GNUプロジェクト[注1]が管理・開発を行っているC言語用のソースコードフォーマッタである．このソフトウェアは，GNUのドキュメント[注2]によると1976年にBSD UNIXの一部として開発され，その後Free Software Foundation[注3]に寄付されたとのことで，非常に長い歴史をもつソフトウェアである．Free Software Foundationに寄付された後，何人かのメンテナが開発を引き継ぎ，現在はDavid Ingamells氏によってメンテナンスが行われている．

インストール

● UNIX

RedHat LinuxなどのLinuxディストリビューションでは，GNU indentは標準でインストールされることも多い．コンソール上で以下のように入力すれば，GNU indentがインストールされているか，インストールされているバージョンはどれかを知ることができる．

```
$ indent --version
GNU indent 2.2.8a      ←現在のバージョン
```

インストールされていない場合は，Webページからソースファイルをダウンロードしてきて，自分でビルドを行う必要がある．

とはいっても，ビルドの方法は非常に一般的なものだ．ソースコードはtarball(.tar.gzファイル)として配布されているので，それをダウンロードして展開する．そして展開されたディレクトリに移動し，以下のようにmakefileを作成し，ビルド，インストールを行うだけである．

```
$ ./configure       ←makefileの生成
$ make              ←プログラムのビルド
$ su                ←パスワードを入力
# make install      ←ファイルのインストール(管理者
                      権限で)
```

標準では，/usr/local/bin/にファイルが置かれる．

● Windows

GNUのソフトウェアのWindowsへの移植を行っている「GnuWin32」というプロジェクトによって，GNU indentのWindows版が作成されている．そのファイルは同プロジェクトのWebページから入手が可能である．バージョンは，残念ながら最新バージョンを確実に追従しているわけではないが，それほど古くないバージョンを入手できる(執筆時，公開されているバージョンは2.27)．

● GnuWin32プロジェクト
http://gnuwin32.sourceforge.net/

ここでは，コンパイル済みのファイルが配布されているので，ダウンロードして，適当な場所で展開するだけである．とくにインストーラなどは利用しない．展開したディレクトリ内の，binディレクトリにあるindent.exeが実行ファイルである．

また，GNU indentはGNU gettextという国際化のためのパ

注1：GNUは「GNU is Not Unix」の略．完全なフリーウェアとしてUNIXの互換システムを作るプロジェクトである．1984年にRichard Stallman氏によって開始された．

注2：http://www.gnu.org/brave-gnu-world/issue-35.ja.html

注3：GNUプロジェクトのための資金を集める団体．代表はRichard Stallman氏．

〔リスト 1〕インデントがめちゃくちゃなソースコード

```
/* 曜日を計算する */
int dayofweek(int y,
  int m,
      int d)
{
  static int t[] = {0, 3, 2, 5,0,3,    5, 1, 4, 6, 2, 4};
      y -= m < 3;
  return (y + y/4 - y/100 + y/400 + t[m-1] + d) % 7;
}

    int main(int argc, char * argv[])
{   int i;
    i = dayofweek
  ( 2003, 1, 1 );
      printf( "2003/1/1 ... %d", i);
  return 0;
}
```

〔リスト 2〕GNU indentによって綺麗に整形されたソースコード

```
/* 曜日を計算する */
int
dayofweek (int y, int m, int d)
{
  static int t[] = { 0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4 };
  y -= m < 3;
  return (y + y / 4 - y / 100 + y / 400 + t[m - 1] + d) % 7;
}

int
main (int argc, char *argv[])
{
  int i;
  i = dayofweek (2003, 1, 1);
  printf ("2003/1/1 ... %d", i);
  return 0;
}
```

ッケージを利用しているので，こちらもインストールする必要がある．以下のページで配布されているlibintl.dllをindent.exeと同じディレクトリ(もしくはシステムディレクトリ)に置けばよい．

● gettext for Win32

http://sourceforge.net/projects/gettext

ソースコードのフォーマット

GNU indentのもっとも単純な使い方は以下のように，単に処理したいソースコードをパラメータとして指定して，呼び出す方法である．

```
indent test.c
```

この場合，test.cというソースコードを標準のフォーマット(GNUスタイル)でフォーマットしなおし，元のファイルを上書きする．

次のように-oオプションを使うことで，出力先を指定することもできる．

```
indent test.c -o test_indent.c
```

たとえば，**リスト1**のようなめちゃくちゃなフォーマットのソースコードであっても，**リスト2**のように，すっきりとしたソースコードにしてくれるのである．

さてGNU indentでは，どういうフォーマットに変換するかをさまざまに設定することができる．そして，フォーマットの設定には，やはりオプションを利用する．ソースコードのフォーマットを指定するオプションは，**表1**のようにさまざまなものが用意されている．

GNU indentにはベースとなる三つの一般的なスタイル(**表1**では最後に記した三つ．詳しくは次節で述べる)が用意されており，それにさらにオプションを追加して，スタイルを修正する形で指定する．

ベースとなるスタイルを指定していない場合は，GNUスタイルが標準で選択される．たとえば，GNUスタイルでは，if文の中括弧({)は以下のように，if文の次の行に書かれる．

〔表 1〕GNU indentで利用できるソースコード整形用のオプション

オプション		意　味
-bad	--blank-lines-after-declarations	定義ブロックの後に空行を入れる
-bap	--blank-lines-after-procedures	プロシージャ(関数)の後に空行を入れる
-bbb	--blank-lines-before-block-comments	コメントブロックの前に空行を入れる
-bbo	--break-before-boolean-operator	論理演算子(&&や||)の前で改行する
-bc	--blank-lines-after-commas	変数定義文でのコンマの後に改行を入れる
-bfda	--break-function-decl-args	関数定義で，カンマの後に改行を入れる
-bl	--braces-after-if-line	「{」をifの次の行に書く
-bli[n]	--brace-indent[n]	「{」の行で何文字インデントするかを指定
-bls	--braces-after-struct-decl-line	「{」をstructの次の行に書く
-br	--braces-on-if-line	「{」をifと同じ行に書く
-brs	--braces-on-struct-decl-line	「{」をstructと同じ行に書く
-bs	--blank-before-sizeof	sizeofの後にスペースを入れる．--Bill-Shannonも可
-c[n]	--comment-indentation[n]	コードの後につけたコメントを何文字インデントするか(標準は33文字)
-cbi[n]	--case-brace-indentation[n]	caseブロックにおける「{」を何文字インデントするか
-cd[n]	--declaration-comment-column[n]	定義文の後につけたコメントを何文字インデントするか(標準は33文字)
-cdb	--comment-delimiters-on-blank-lines	コメントデリミタ(「/*」と「*/」)を単独の行にする
-cdw	--cuddle-do-while	do-whileループで，「}」とwhileを同じ行に書く
-ce	--cuddle-else	elseと一つ前の「}」を同じ行に書く
-ci[n]	--continuation-indentation[n]	一つの文が複数行にわたる場合の，2行目以降のインデントを指定

〔表1〕GNU indentで利用できるソースコード整形用のオプション（つづき）

オプション		意　味
-cli[n]	--case-indentation[n]	case文を何文字インデントするか
-cp[n]	--else-endif-column	「#else」「#endif」の後ろにつけたコメントを何文字インデントするか（標準は33文字）
-cs	--space-after-cast	キャストオペレータの後にスペースを入れる
-d[n]	--line-comments-indentation[n]	コメントのみの行を何文字インデントするかを指定
-di[n]	--declaration-indentation[n]	変数定義文で，変数型と変数名のあいだにいくつスペースをあけるかを指定
-fc1	--format-first-column-comments	行の1文字目から始まるコメントをフォーマット
-fca	--format-all-comments	行の2文字目以降から始まるコメントをフォーマット
-hnl	--honour-newlines	すでにある改行位置を尊重する
-i[n]	--indent-level[n]	一回のインデントの文字数を指定
-ip[n]	--parameter-indentation[n]	括弧の数に応じてインデントする文字数を指定
-l[n]	--line-length	1行の最大文字数を指定（コメント行を除く）
-lc[n]	--comment-line-length	コメント行1行の最大文字数を指定
-lp	--continue-at-parentheses	括弧が閉じられずに改行された場合，次の行は括弧の位置までインデントされる
-lps	--leave-preprocessor-space	プリプロセッサの「#」とコマンド名のあいだにスペースがあっても取り除かない
-nbad	--no-blank-lines-after-declarations	定義ブロックの後に空行を入れない
-nbap	--no-blank-lines-after-procedures	プロシージャ（関数）の後に空行を入れない
-nbbo	--break-after-boolean-operator	論理演算子（&&や||）の後で改行する
-nbc	--no-blank-lines-after-commas	変数定義文でのカンマの後に改行を入れない
-nbfda	--dont-break-function-decl-args	関数定義で，カンマの後に改行を入れない
-ncdb	--no-comment-delimiters-on-blank-lines	コメントデリミタ（「/*」と「*/」）を単独の行にしない
-ncdw	--dont-cuddle-do-while	do-whileループで，「}」の次の行にwhileを書く
-nce	--dont-cuddle-else	elseを一つ前の「}」の次の行に書く
-ncs	--no-space-after-casts	キャストオペレータの後にスペースを入れない
-nfc1	--dont-format-first-column-comments	行の1文字目から始まるコメントをフォーマットしない
-nfca	--dont-format-comments	コメントのフォーマットを行わない
-nhnl	--ignore-newlines	もとからあった改行位置を無視する
-nip	--no-parameter-indentation	-ip0と等価．括弧の数に応じたインデントを行わない
-nlp	--dont-line-up-parentheses	括弧が閉じられずに改行された場合でも，次の行のインデントはそれとは無関係
-npcs	--no-space-after-function-call-names	関数名と括弧の間にスペースを入れない
-npro	--ignore-profile	設定ファイルを無視する
-nprs	--no-space-after-parentheses	括弧の後にスペースを入れない
-npsl	--dont-break-procedure-type	関数の定義において変数名と，その関数の戻り値の型の間に改行を入れない
-nsaf	--no-space-after-for	forとそれに続く括弧の間にスペースを入れない
-nsai	--no-space-after-if	ifの後に空白をあけない
-nsaw	--no-space-after-while	whileの後に空白をあけない
-nsc	--dont-star-comments	コメントの左側にアスタリスクを挿入しない
-nsob	--leave-optional-blank-lines	余分な空行を削除しない
-nss	--dont-space-special-semicolon	forやwhileなどで使われるセミコロンの前に，スペースを挿入しない
-nut	--no-tabs	スペースをタブに置き換えない
-pcs	--space-after-procedure-calls	関数名と括弧の間にスペースを入れる
-pi[n]	--paren-indentation	「(」で始まる行で何文字インデントするか
-prs	--space-after-parentheses	「(」の前，および「)」の後にスペースを入れる
-psl	--procnames-start-lines	関数の定義において変数名と，その関数の戻り値の型のあいだに改行を入れなる
-saf	--space-after-for	forとそれに続く括弧の間にスペースを入れる
-sai	--space-after-if	ifの後に空白をあける
-saw	--space-after-while	whileの後に空白をあける
-sbi[n]	--struct-brace-indentation[n]	struct，union，enumの行を何文字インデントするかを指定
-sc	--start-left-side-of-comments	コメントの左側にアスタリスクを挿入
-sob	--swallow-optional-blank-lines	余分な空行を削除する
-ss	--space-special-semicolon	forやwhileなどで使われるセミコロンの前に，スペースを挿入する
-ts[n]	--tab-size[n]	一つのタブに置き換えられるスペースの数を指定する
-ut	--use-tabs	スペースをタブに置き換える
-gnu	--gnu-style	GNUスタイルに指定
-kr	--k-and-r-style	K&Rスタイルに指定
-orig	--original	Berkeleyスタイルに指定

```
if( flag == 1 )
  {
    return -1;
  }
```

しかし，GNUスタイルをベースとするが，if文の中括弧はif文と同じ行に書きたいといった場合は，以下のように-brというオプションを指定すればいい．

```
$ indent -br test.c
```

こうすれば，以下のようにif文の中括弧はif文と同じ行に置かれるようになる．

```
if( flag == 1 ) {
    return -1;
  }
```

また，インデントの文字数(GNUスタイルでは2文字)を4文字にして，なおかつタブへの置き換えをしないようにするには，以下のように指定する．

```
$ indent -i4 -nut test.c
```

標準では，8文字のスペースがあると，タブに置き換えられるようになっているが，-nutを指定することで，タブへの置き換えを無効にすることができる(何文字でタブに置きかえるかは，-tsというオプションで指定することができる)．

なお，スタイルを指定するオプションは，左から右へと解釈される．相反するスタイル(たとえば-badと-nbad)が両方指定されている場合には，より右側で指定したものが有効になる．

GNU indentの使い方のポイントは，これらのコマンドをうまく使い分け，いかに自分のスタイルに近い組み合わせを作るかという点にある．

なお，GNU indentは国際化が行われており，コメント，文字列などに日本語が入っていても，とくに問題なく利用することができる[注4]．

GNU indentがサポートするC言語の一般的なスタイル

人によってさまざまなスタイルが存在するC言語の記述方法だが，一般的によく用いられるスタイルというものが存在する．GNU indentでは三つの「標準的な」スタイルをサポートしており，それらのスタイルは個々のオプションを細かく指定しなくても，ひとつのオプションで簡単に指定できるようになっている．

サポートされているスタイルは「GNUスタイル」，「Kernighan & Ritchieスタイル」，「Berkeleyスタイル」の3種類である．

● GNUスタイル

このスタイルは，GNUコーディング規約[注5]で推奨されているもので，現在のGNU indentの標準スタイルである．このスタイルを指定するために，「-gnu」というオプションが用意されているが，実際には標準で指定されるスタイルのため，このオプションを使う必要はない．

GNUスタイルを，個々のスタイル指定で表すと，以下のようになる．

```
-nbad -bap -nbc -bbo -bl -bli2 -bls -ncdb -nce
-cp1 -cs -di2 -ndj -nfc1 -nfca -hnl -i2 -ip5
-lp -pcs -nprs -psl -saf -sai -saw -nsc -nsob
```

● Kernighan & Ritchieスタイル

C言語は1973年，(当時)AT&Tのベル研究所において，Brian W. Kernighan氏とDennis M. Ritchie氏によって言語仕様が作られた．両氏は現在でもC言語の原典として広く読まれつづけている『プログラミング言語C』の著者でもある．そしてこの書籍の中で用いられているスタイルが，Kernighan & Ritchieスタイルである．

このスタイルは「-kr」というオプションで指定できる．個々のスタイル指定で表すと，以下のようになる．

```
-nbad -bap -bbo -nbc -br -brs -c33 -cd33 -ncdb
-ce -ci4 -cli0 -cp33 -cs -d0 -di1 -nfc1 -nfca
-hnl -i4 -ip0 -l75 -lp -npcs -nprs -npsl -saf
-sai -saw -nsc -nsob -nss
```

● Berkeleyスタイル

GNU indentはもともとBSD UNIXの一部として開発されており，その当時の標準のスタイルが「Berkeleyスタイル」と呼ばれている．そのため，このスタイルは「-orig(オリジナルの意味)」というオプションで指定する．個々のスタイル指定で表すと，以下のようになる．

```
-nbad -nbap -bbo -bc -br -brs -c33 -cd33 -cdb
-ce -ci4 -cli0 -cp33 -di16 -fc1 -fca -hnl -i4
-ip4 -l75 -lp -npcs -nprs -psl -saf -sai -saw
-sc -nsob -nss -ts8
```

設定をファイルで指定する

GNU indentでは，そのフォーマットスタイルを実行時に指定できるほか，あらかじめ設定ファイルを記述しておいてその設定を参照させることができる．自分のスタイルをあらかじめファイルに記述しておくことで，フォーマットを毎回指定しなくても簡単に自分のスタイルに修正することが可能になる．

設定を記述するファイルは.indent.proという名前にする．そしてカレントディレクトリに置いておくと，GNU indentが自動的に読み込んでくれる．

設定ファイルには，**表1**で示したオプションを改行やスペー

注4：ただし，国際化が行われたのは，2.2.7からで，それ以前のバージョンではシフトJISの文字列が入っているとうまく動かないなどの問題があった．そのため，シフトJISに対応した修正版なども公開されていたが，現在ではその必要もなくなっている．

注5：http://www.sra.co.jp/wingnut/standards-j_toc.html

〔リスト3〕.indent.pro サンプル

```
/*
.indent.pro サンプル
*/
--k-and-r-style
--blank-lines-after-declarations
--braces-after-if-line
--no-tabs
/*
--swallow-optional-blank-lines     ←コメントアウトされている
*/
```

スで区切って記述すればよいだけである．設定ファイルでは，C/C++ のコメントスタイルである「//」および「/* 〜 */」を使って，設定をコメントアウトすることもできる（**リスト3**）．

-npro（--ignore-profile）というオプションを指定することで，実行時に設定ファイルを読み込まなくすることも可能である．

C言語以外のプログラムをフォーマットするには

GNU indent の弱点は，対応している言語がC言語のみという点である．GNU indent の Web サイトでは，C++ はサポートしていないということが強調されている．したがって，C++ をはじめとする他の言語を利用する場合には，何か他のフォーマットツールを利用する必要がある．

以下の「C-C++ Beautifier HOW-TO」というドキュメントには，C/C++ をはじめ，Perl や Java などさまざまな言語向けのソースコードフォーマッタが紹介されているので参考になるだろう．

- C-C++ Beautifier HOW-TO
  http://www.linuxselfhelp.com/howtos/C-C++Beautifier/C-C++Beautifier-HOWTO.html

また，GNU indent を C++ に対応させようとするプロジェクトも開始されている．まだ2002年7月に開始されたばかりのプロジェクトで，何の成果物も公開されてはいないが，今後 C++ に対応した GNU indent（Indent++）が使える日がくるかもしれない．

- Indent++
  http://www.uvm.edu/~ashawley/indent++/

Column　COBF

ソースコードフォーマッタは，かならずしもソースコードを読みやすくするとは限らない．GNU indent を使っていても，自分のローカルルールと異なるルールに変換してしまえば，（自分には）読みづらいソースとなる．

しかし，そもそも読みやすいソースを作ることを目的としないフォーマッタも存在するのである．そんなフォーマッタの一つであるCOBFをここで紹介する．これは，GNU indent のようなフォーマッタとは正反対の機能をもったソースコードフォーマッタである．つまりCOBFは，ソースコードを「読みにくくする」ことを目的としたフォーマッタなのである．

たとえば，**リストA**のようなソースコードがあったとする．これを，COBFで変換すると，**リストB**のようになってしまう．

なんとも読みづらい，というよりもまったく読むことができないソースコードに変換されてしまうことがわかるだろう．しかも，COBFで変換したソースコードは，人間には非常に読みづらいものにはなるが，きちんとコンパイルすることが可能なのだ．

そのため，ソースコードを公開しなければならないが，あまり内容を解析されたくないといった場合に利用するのがよいそうである（ドキュメントにそう書かれている）．

名称：COBF（C-Obfuscator）
作者：Bernhard Baier 氏
Web サイト：http://home.arcor.de/bernhard.baier/cobf/
現在のバージョン：1.03

- コードを読みにくくする手法

COBFは，コードを読みにくくするために，以下の方法を用いる．

1) コメントをすべてなくす
2) 改行をなくす
3) 変数名を無意味なものに置き換える
4) 予約語を無意味なものに置き換える
5) 文字列を16進数に置き換える

- コメントをすべてなくす

これは，文字どおりすべてのコメントが削除されてしまうことを意味する．コメントはソースコードのどこでどんな処理が行われているかといったことを知るための重要な手がかりとなる．そのコメントを削除することで，どこで何が行われているかが，ソースを解読しないことにはわからなくなる．

- 改行をなくす

GNU indent では，どこに改行を入れるかで，見やすさを調整していた．改行を取り除いてしまうことで，ソースコードは読みづらくなる．

- 変数名を無意味なものに置き換える

ソースコードを読みやすくするには，わかりやすい変数名を使うことも重要である．ということはつまり，変数名を規則性のないものに代えてしまうことで，ソースコードが読みづらくなるのである．

- 予約語を無意味なものに置き換える

「if」，「for」，「switch」といった予約語を，「#define」で「a」，「b」といったまったく無意味な文字列に置換してしまう．これは，COBFの行う処理の中でももっとも効果的なものである．これを行うと，その文字が何を意味するのかがわからなくなり，また変数名との区別もなくなってしまうので，ソースコードは著しく読みにくくなる．

この置き換えを行うために，COBFは cobf.h という新しいイン

まとめ：ソースコードフォーマッタの必要性

オープンソースという言葉を，最近よく耳にするようになった．これは，ソフトウェアのソースコードを公開してしまう開発スタイルのことである．SourceForge[注6]というオープンソース開発者のためのホスティングサービスや各種ツールを公開しているサイトには，オープンソースで開発されているプロジェクトがたくさん登録されているし，Mozilla.orgやOpenOffice.orgなど，もともとは市販されていたソフトウェアが，オープンソースになったものある．

オープンソースで開発を行うということは，(あたりまえだが)ソースコードを第三者が読むことができるという意味である．しかし，せっかくソースコードが公開されていても，自分以外の書いたソースコードというのは，非常に読みづらいことが多い．その原因には，コメントの有無や，プログラム自体の流れがすっきりしているかどうかなども挙げることができるが，そのほかに，改行をどこで入れるのか，インデントをどうするのかといったプログラマそれぞれがもつローカルルールの違いによって生じる読みづらさも存在する．ソースコードフォーマッタは，そういった自分とは違うルールで書かれた，(自分には)読みづらいソースに出会ったときに威力を発揮するはずだ．

また，自分で作成したプログラムを公に公開する場合でも，そのスタイルを一般的によく利用されているものに変換して，万人に読みやすいプログラムにしてから公開することもできる．

このように，ソースコードフォーマッタは，ソースコードを共有する際には，非常に強力な武器となってくれるツールなのである．

注6：http://sourceforge.net/，日本語版であるhttp://sourceforge.jp/もある．

みずの・たかあき

クルードファイルを生成し，そこにたとえば以下のように#define文を定義している．

```
#define c int
#define e main
#define d for
#define f printf
#define g return
```

この例だと，forは「d」に，printfは「f」に置き換えられてしまう．

●文字列を16進数に置き換える

C/C++では，文字列を16進数で記すことができる．こうすると，元の文字に戻さないと何が書かれているかがわからないので，ソースコードを読みづらくすることができる．

```
printf("H");
  ↓
printf("¥x48");
```

このように，COBFは，ソースコードを非常に読みづらくすることができる．もちろん，まったく読めなくなるということではなく，予約語を元に戻し，GNU indentなどでフォーマットしなおすことで，ある程度読むことができるようになる．したがって，どれくらいの有効性があるのかは疑問だが，すぐには読めなくなるし，面倒で読む気をなくさせるという意味では，非常に有効かもしれない．

そもそも，ソースを読みづらくするということにどれだけの意義があるのか，それはよくわからないが，こうしたヘンテコなアイデアをまじめに形にするということもプログラミングの醍醐味の一つではないだろうか．

〔リストA〕整形前のソースリスト

```
#include <stdio.h>
#ifdef unix
#define MAX COUNT 10
#else
#define MAX COUNT 20
#endif

int main()
{
    int i;
    for (i = 0; i < MAX COUNT; ++i)
            printf("Hello %d!¥n", i);
    return 0;
}
```

〔リストB〕整形後のソースリスト

```
/* COBF by BB -- 'test.c' obfuscated at Fri Oct 11 10:30:31 2002
*/
#include<stdio.h>
#include"cobf.h"
#ifdef unix
#define b 10
#else
#define b 20
#endif
c e(){c a;d(a=0;a<b;++a)f("¥x48¥x65¥x6c¥x6c¥x6f¥x20¥x25¥x64¥x21¥n",a);
g 0;}
```

第6回

GCCのインストールとC言語におけるGCCの拡張機能

岸 哲夫

今回は，GCCを各種環境にインストールする際の注意点を解説し，続けてGCCで拡張されているC言語の仕様について，今回と次回の2回に分けて説明する．具体的には，今回は，

- 式の中の文と宣言
- ローカルに宣言されたラベル
- 値としてのラベル
- 入れ子になった関数
- 関数呼び出しの構築
- 型の代入
- typeofでの型の参照
- 拡張されたlvalue

について，解説を行う．

（編集部）

GCCの導入に際して

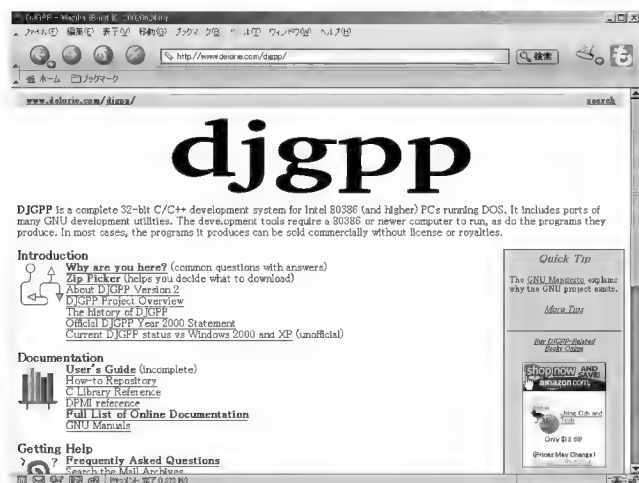
現在，インテルアーキテクチャで動作するLinuxマシンにGCCを導入することは簡単です．実際，インストールのためのパッケージやバイナリが数多く配布されています．それらを利用することで，とくに悩む必要はなくなりました．

それでも通常のインストールが必要な環境の場合もあるので，詳細を説明をします．

● MS-DOSの場合

本連載では，基本的にLinuxで動作するGNUツールを扱っていく予定ですが，他の環境で動作するGCCについて少し説明しておきます．

「素の」MS-DOSはあまり使われなくなってきていると思いますが，組み込み用途などではまだ活躍しています．この環境でGCCを動作させるなら，DJGPPというツールをインストールしなくてはなりません．このセットには，DOS-Extenderツールが含まれているので，32ビット処理が可能になります．

公式サイトを**図1**に示します．

また，特殊な用途になりますが，「8086」プロセッサ環境の16ビットMS-DOSにもインストール可能です．**図2**に示すWebページからダウンロードできます．

もちろん，LinuxなどのGCCとそのまま同じではなく，以下のような制約があります．

- `far`ポインタが使用できない
- `double`および`float`が使用できない
- コードが64Kバイト，データが64Kバイト以上使用できない
- C++の`constructors`と`destructors`が使用できない
- デバッグツールがない
- ライブラリがない．また，スタートアップコードも標準のヘッダもない

この場合，リンカもDJLINKというツールを使います．

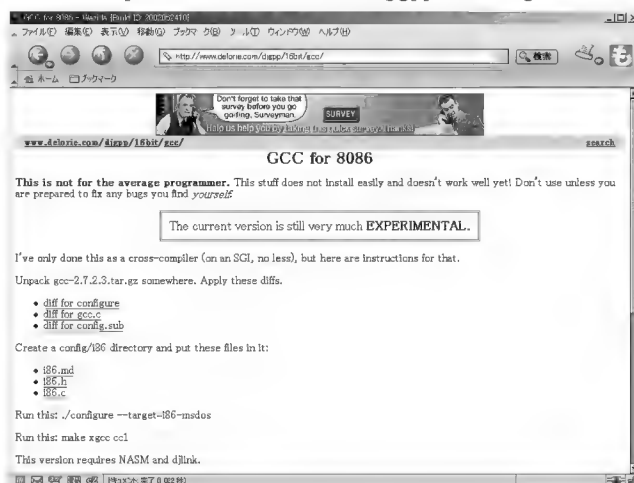
● BeOSの場合

インストールした時点でGCCが導入されているはずです．バージョンを上げる場合には，元のGCCでコンパイルが可能です．

● Mac OS Xの場合

やはり最初からGCCが導入されています．CCという名前で/usr/bin/ccにあります．

〔図1〕djgppのWebページ(http://www.delorie.com/djgpp/)

〔図2〕gcc for 8086
(http://www.delorie.com/djgpp/16bit/gcc/)

● VMSシステム

GCCのVMSバージョンは，バックアップ・セーブセットとして配布されています．ソースコードとコンパイル済みのバイナリが含まれています．

しかし，まだ多くのバグがあるようです．それを回避する策の一つは，導入した環境でバイナリを再作成することです．

● Sunにインストールする場合の注意

Solarisでは，GCCをビルドする際，/usr/ucbにあるリンカやその他のツールを使わず，/usr/ccs/binのほうにあるツールを使ってください．ビルド中にアセンブラでエラーを発生する場合があります．

その場合，最新のGNUアセンブラを使用してください．binutilsの最新版にアップグレードするか，またはSolarisのアセンブラを使ってください．

libgcc.aをコンパイルするときには，環境変数FLOAT_OPTIONがセットされていないことを確認してください．libgcc.aのコンパイル時にこのオプションがf68881にセットされていると，生成されるコードは特殊なスタートアップファイルとリンクすることを要求され，正しくリンクできなくなります．

また，Sunのライブラリのバージョンによっては，allocaにバグがあります．このバグを回避するには，GCCによってコンパイルされたGCCバイナリをインストールしてください．これは，allocaを組み込み関数として使います．ライブラリの中の関数を使うことはありません．

Sunのコンパイラのバージョンによっては，GCCをコンパイルする際にクラッシュします．問題になるのは，cppにおけるセグメンテーションフォルトです．この問題の原因は，環境変数に大量のデータが存在することにあるように思われます．SunのCCを使ってGCCをコンパイルするのに以下のコマンドを使うことによって，この問題を回避することができるかもしれません．

```
make CC="TERMCAP=x OBJS=x LIBFUNCS=x
                                STAGESTUFF=x cc"
```

SunOS 4.1.3と4.1.3_U1にはバグがあるようで，GCCをコンパイルする際に問題が起きます．その場合，最新のOSにアップグレードしてください．

ざっと説明しましたが，このようにMS-DOSでもMac OS XでもGCCを使用することは可能です．また，VMS環境でも，もちろん商用UNIX環境にも導入が可能です．場合によっては，そのマシンで使用されるオリジナルな環境よりも効率が良いかもしれません．

C言語におけるGCCの拡張機能

ここではC言語に関することだけ扱います．C++，Objective Cに関しては，回を改めて説明することにします．

GNUのCには，ANSI規格にしたがっていない拡張機能がいくつかあります．たしかに便利なのですが，それに慣れてしまうと他の環境でプログラムが書けなくなってしまう恐れがあります．また，他の環境にポーティングすることが難しくなってしまいます．

しかし，用途によっては拡張機能によってわかりやすくコーディングすることが必要かもしれません．したがって，どのように使用するかで拡張機能を排除するか許可するかを考えてください．

もし排除する場合は，連載第3回目(2002年10月号)に記したように，-pedanticオプションを使用します．

このオプションは厳密なANSI CおよびISO C++により要求される警告をすべて出力します．禁止されている拡張機能を使うプログラムをすべて拒絶します．正当なANSI Cプログラム，およびISO C++プログラムであれば，このオプションの指定の有無にかかわらず正しくコンパイルされるはずです．しかし，このオプションが指定されないと，特定のGNU拡張機能もまたサポートされることになります．

もし，拡張機能を使用してコードを書いた場合，条件コンパイルでANSI，非ANSIを分けることが可能です．これらの機能が利用可能であるかどうかをテストするためには，__GNUC__というマクロが事前に定義されているかどうかをチェックします．この__GNUC__というマクロは，GCCでは常に定義されています．

LinuxのGUI開発環境として話題のC++言語版のKylixでは，GCCの独自拡張機能を使う場合，新たに環境の設定をしなくてはなりません．もし，Kylixとの互換を取るならば，ANSI準拠にしなくてはなりません．

今回と次回の2回にわたり，この拡張機能の説明と検証を行います．

● 式の中の文と宣言

拡張機能として，小括弧()内に中括弧{}でまとめた複文を入れることができます．この複文中ではループや変数宣言もできます．最後の項がやはり小括弧()の値を表します．

引き数のうち大きなほうを戻すマクロmax_a()を，次のように作ることができます．

```
#define max_a(X,Y) ¥
    ({int _X = (X), _Y = (Y); _X > _Y ? _X :
                                      _Y; })
```

引き数の値を一時的な変数に保管し，それの大小を評価しているので，「マクロの副作用」にはまることはありません．

GCCの標準機能で実装するには，inline関数を使うことです．そのほうが確実で簡単かつ安全です(**リスト1**，**リスト2**)．

実行結果は以下のようになります．

```
$ ./test44
副作用が発生した答え = 2
正しい答え(拡張機能) = 80
$ ./test45
副作用が発生した答え = 2
```

〔リスト1〕副作用が発生するマクロ，および拡張機能を使ったマクロを使用したCソース(test44.c)

```
#include <stdio.h>
#define max(X, Y) X > Y ? X : Y
#define max_a(X,Y) ¥
    ({int _X = (X), _Y = (Y); _X > _Y ? _X : _Y; })
int main(void)
{
    int  a,b,c,ans;
    a = 2;
    b = 10;
    c = 8;
    ans = c * max(a, b);
    printf("副作用が発生した答え = %d¥n", ans);
    ans = c * max_a(a, b);
    printf("正しい答え（拡張機能） = %d¥n", ans);
    return;
}
```

```
正しい答え(拡張機能) = 80
正しい答え1(インライン関数) = 80
正しい答え2(インライン関数) = 160
$
```

リスト3にtest44.cのアセンブラリストを，**リスト4**にtest44.cのアセンブラリストを示します．

リストからわかるように，ここではinline関数を指定しても展開されません．C++の処理を行っていないからです．

リスト1の二つのマクロは，以下のように展開されます．

```
ans = c * a > b ? a : b ;
ans = c * ({int _X = ( a ), _Y = ( b );
                          _X > _Y ? _X : _Y; })
```

上の式では問題が起きることがわかると思います．

● ローカルに宣言されたラベル

ローカルラベル機能が役に立つのは，複文式〔小括弧()内に中括弧{}で文をまとめたもの〕がマクロの中でしばしば使われるからです．あるマクロの中に入れ子になったループがあると，そのループから抜け出るのにgotoが役に立ちます．ただし，その場合，スコープが関数全体である通常のラベルを使うことはできません．

なぜなら，そのマクロが一つの関数の中で複数回展開されることがあると，同じ関数の中でそのラベルが複数回定義されることになるからです．局所ラベルはこのような問題を回避します．

GCCの標準機能で実装するには，ラベルを使用する命令を使わないことです．プログラムは冗長になるかもしれませんが，可読性は高まります．

● 値としてのラベル

これは昔からメインフレームのCOBOLやアセンブラやROM BASICでコードを小さくするために使用されてきた「飛び先テーブル」の一種です．

これを使用すると読みにくく，バグを出したときに見つけにくいので使用しないほうがよいと思います．

そのラベルがスコープ内にある関数の中で定義されたラベルのアドレスを，単項演算子&&で獲得することができます．この値の型はvoid *です．この値は定数であり，void *型の定数

〔リスト2〕リスト1に加えてinline関数を使用したCソース(test45.c)

```
#include <stdio.h>
#define max(X, Y) X > Y ? X : Y
#define max_a(X,Y) ¥
    ({int _X = (X), _Y = (Y); _X > _Y ? _X : _Y; })
static __inline__ int max_i(int X,int Y);
int main(void)
{
    int  a,b,c,ans;
    a = 2;
    b = 10;
    c = 8;
    ans = c * max(a, b);
    printf("副作用が発生した答え = %d¥n", ans);
    ans = c * max_a(a, b);
    printf("正しい答え（拡張機能） = %d¥n", ans);
    ans = c * (    max_i(a, b)    );
    printf("正しい答え1（インライン関数） = %d¥n", ans);
    ans = c * max_i(a, b) * 2;
    printf("正しい答え2（インライン関数） = %d¥n", ans);
    return;
}
static __inline__ int max_i(int X,int Y)
{
    return X > Y ? X : Y;
}
```

が正当であるところではどこでも使うことができます．次に例を示します．

```
static void *array[]
                = { &&foo, &&bar, &&hack };
```

こうすると，以下のようにインデックスを使ってラベルを選択することができます．

```
goto *array[i];
```

ここで，配列の添字が上限内にあるかどうかチェックされないことに注意してください．Cにおける配列のインデックスでは，このようなチェックは決して行われません．

このようなことを行うのならば，switch文のほうが，よりすっきりしています．実現しようとしていることがswitch文にうまく適合しない場合以外は，ラベル値の配列ではなく，switch文を使うようにしてください．

● 入れ子になった関数

C言語の拡張機能でネストされた関数を作ることができます(**リスト5**)．C++言語では使えません．

入れ子関数が定義された箇所において可視な変数は，すべて入れ子関数の中からアクセスすることができます．

関数の中で変数定義ができるところであればどこでも，入れ子関数の定義を行うことができます．つまり，任意のブロック内で，そのブロックの最初にある文の前であれば，入れ子関数の定義が可能です．

入れ子関数のアドレスをどこかに格納したり，別の関数へ渡すことによって，入れ子関数の名前が有効なスコープの外部からでも，入れ子関数を呼び出すことができます(**リスト6**)．

リスト6の関数test1はstoreのアドレスを引き数として受け取っています．test1がstoreを呼び出すと，storeに渡された引き数はarrayへの値の格納に使われます．しかし，このテクニックは，入れ子関数を包含している関数(この例では

〔リスト 3〕**test44.c のアセンブラリスト**（test44.s）

```
    .file     "test44.c"
    .version  "01.01"
gcc2 compiled.:
.section .rodata
.LC0:
    .string   "¥311¥373¥272¥356¥315¥321¥244¥254¥310¥257¥300¥270¥244¥267¥244¥277¥305¥372¥244¥250 = %d¥n"
.LC1:
    .string   "¥300¥265¥244¥267¥244¥244¥305¥372¥244¥250¥241¥312¥263¥310¥304¥245¥265¥241¥307¥275¥241¥313 = %d¥n"
.text
    .align 4
.globl main
    .type      main,@function
main:
    pushl %ebp
    movl %esp,%ebp
    subl $40,%esp
    movl $2,-4(%ebp)
    movl $10,-8(%ebp)
    movl $8,-12(%ebp)
    movl -12(%ebp),%eax
    imull -4(%ebp),%eax
    cmpl -8(%ebp),%eax
    jle .L3
    movl -4(%ebp),%eax
    jmp .L4
    .p2align 4,,7
.L3:
    movl -8(%ebp),%eax
.L4:
    movl %eax,-16(%ebp)
    addl $-8,%esp
    movl -16(%ebp),%eax
    pushl %eax
    pushl $.LC0
    call printf
    addl $16,%esp
    movl -4(%ebp),%eax
    movl %eax,-20(%ebp)
    movl -8(%ebp),%eax
    movl %eax,-24(%ebp)
    movl -24(%ebp),%eax
    cmpl -20(%ebp),%eax
    jge .L5
    movl -20(%ebp),%eax
.L5:
    imull -12(%ebp),%eax
    movl %eax,-16(%ebp)
    addl $-8,%esp
    movl -16(%ebp),%eax
    pushl %eax
    pushl $.LC1
    call printf
    addl $16,%esp
    jmp .L2
    .p2align 4,,7
.L2:
    movl %ebp,%esp
    popl %ebp
    ret
.Lfe1:
    .size      main,.Lfe1-main
    .ident    "GCC: (GNU) 2.95.3 20010315 (release)"
```

〔リスト 4〕**test45.c のアセンブラリスト**（test45.s）

```
    .file     "test45.c"
    .version  "01.01"
gcc2 compiled.:
.section .rodata
.LC0:
    .string   "¥311¥373¥272¥356¥315¥321¥244¥254¥310¥257¥300¥270¥244¥267¥244¥277¥305¥372¥244¥250 = %d¥n"
.LC1:
    .string   "¥300¥265¥244¥267¥244¥244¥305¥372¥244¥250¥241¥312¥263¥310¥304¥245¥265¥241¥307¥275¥241¥313 = %d¥n"
    .align 32
.LC2:
    .string   "¥300¥265¥244¥267¥244¥244¥305¥372¥244¥2501¥241¥312¥245¥244¥245¥363¥245¥351¥245¥244¥245¥363¥264¥330¥277¥364¥241¥313 = 
                                                                                                            %d¥n"
    .align 32
.LC3:
    .string   "¥300¥265¥244¥267¥244¥244¥305¥372¥244¥2502¥241¥312¥245¥244¥245¥363¥245¥351¥245¥244¥245¥363¥264¥330¥277¥364¥241¥313 = 
                                                                                                            %d¥n"
.text
    .align 4
.globl main
    .type      main,@function
main:
    pushl %ebp
    movl %esp,%ebp
    subl $8,%esp
    addl $-8,%esp
    pushl $2
    pushl $.LC0
    call printf
    addl $16,%esp
    addl $-8,%esp
    pushl $80
    pushl $.LC1
    call printf
    addl $-8,%esp
    pushl $10
    pushl $2
    call max i
    sall $3,%eax
    addl $32,%esp
    addl $-8,%esp
    pushl %eax
```

〔リスト4〕**test45.cのアセンブラリスト**(test45.s)(つづき)

```
    pushl $.LC2
    call printf
    addl $-8,%esp
    pushl $10
    pushl $2
    call max_i
    sall $4,%eax
    addl $32,%esp
    addl $-8,%esp
    pushl %eax
    pushl $.LC3
    call printf
    movl %ebp,%esp
    popl %ebp
    ret
.Lfe1:
    .size      main,.Lfe1-main
    .align 4
    .type      max_i,@function
max_i:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%edx
    movl 12(%ebp),%eax
    cmpl %edx,%eax
    jge .L22
    movl %edx,%eax
.L22:
    movl %ebp,%esp
    popl %ebp
    ret
.Lfe2:
    .size      max_i,.Lfe2-max_i
    .ident    "GCC: (GNU) 2.95.3 20010315 (release)"
```

〔リスト5〕**ネストされた関数**

```
void foo( double x[], double y[], int num )
{
  int i;
  int X( double xd )
  {
    return( (int)(2.0*xd) );
  }
  int Y( double yd )
  {
    return( (int)(4.0*yd) );
  }

  for( i=0 ; i<num ; i++ )
  {
    printf("%d %d¥n", X(x[i]), Y(y[i]) );
  }
}
```

test)が終了しないかぎりにおいてのみ有効です.

かりに関数test1内部でどこかにstoreのアドレスを保存しても，関数testが終了した後，そこを参照しても不定です．確実にコアダンプしてしまうでしょう.

GCCの標準機能で同等の機能を実装することはできません．可読性の点から，あまり使わないほうが無難です.

● 関数呼び出しを構築する

以下で説明する組み込み関数を使うことで，ある関数の引き数の数や型がわからなくても，その関数が受け取った引き数を記録して，別の関数を同じ引き数で呼び出すことができます.

また，その関数が返そうとした戻り値のデータ型がわからなくても，その関数呼び出しの戻り値を記録して，後にその値を返すことができます.

○ __builtin_apply_args ()

この組み込み関数は，現在処理中の関数を呼び出すためのデータを返します．関数は，スタックに割り当てられたメモリのブロックへ引き数を関数へ渡すために使われる引き数，ポインタ，レジスタ，構造体値アドレスとすべてのレジスタを保存します．その後，そのデータブロックのアドレスをvoidポインタとして返します.

○ __builtin_apply (function, arguments, size)

この組み込み関数は，引き数(void*型)とサイズ(int型)によって記述されたパラメータのコピーで関数〔void(*)()型〕を

〔リスト6〕**入れ子関数の呼び出し**

```
test (int *array, int size)
{
  void store (int index, int value)
    { array[index] = value; }

  test1 (store, size);
}
```

起動します．引き数の値は上の__builtin_apply_argsによって返された値でなければなりません．引き数サイズはバイトで，スタック引き数データのサイズを指定します.

この関数は，対象の関数が戻す値のデータを返します.

データは，スタックに割り当てられたメモリのブロックで保存されます．その後，そのデータブロックのアドレスをvoidポインタとして返します.

○ __builtin_return (result)

この組み込み関数は，resultの値を戻り値として，この組み込み関数を実行している関数から復帰します．resultには，__builtin_applyから返された値を指定しなければなりません.

GCCの標準機能で同等の機能を実装することはできません.

● 型の代入

初期化子(initializer)とともにtypedef宣言を使うことで，式の型に名前を与えることができます．以下に，式expの型の名前としてnameを定義する方法を示します.

```
typedef name = exp;
```

この機能を使うと先に使ったマクロで，型にとらわれないものが作成できます(**リスト7**).

```
#define max_a(X,Y) ¥
({typedef _X = (X), _Y = (Y);
                    _X > _Y ? _X : _Y; })
```

これで文字列以外の比較ができるようになります．実行結果を以下に示します.

```
$ ./test46
正しい答え(拡張機能) = 80
正しい答え(拡張機能) = 1246822400
$
```

〔リスト7〕マクロに型の代入を使ったソース(test46.c)

```
#include <stdio.h>
#define max_a(a,b) ({ typedef _ta = (a), _tb = (b); _ta _a = (a); _tb _b = (b); _a > _b ? _a : _b; })
int main(void)
{
    int  a,b,c,ans;
    long la,lb,lc,lans;
    a = 2;
    b = 10;
    c = 8;
    la = 2000000;
    lb = 1000000;
    lc = 8000000;
    ans = c * max_a(a, b);
    printf("正しい答え（拡張機能） = %d¥n", ans);
    lans = lc * max_a(la, lb);
    printf("正しい答え（拡張機能） = %d¥n", lans);
    return;
}
```

long値でも正しい結果になりました．GCCの標準機能で同等の機能を実装することはできません．

● typeofで型を参照する

式の型を参照する方法の一つにtypeofがあります．このキーワードを使うときの構文はsizeofの構文に似ていますが，意味論的には，typedefにより定義された型名のような働きをします．

typeofを使う宣言の意味と，なぜそれが役に立つことがあるのかを理解するために，これを次のようなマクロで書き換えてみましょう．

```
#define pointer(T)  typeof(T *)
#define array(T, N) typeof(T [N])
```

すると，上の宣言は次のように書き換えることができます．

```
array (pointer (char), 4) y;
```

この場合，array (pointer (char), 4)は，charへの四つのポインタから構成される配列の型です(**リスト8**，**リスト9**)．実行結果を以下に示します．

```
$ ./test47
char_p[0] = 97
char_p[1] = 98
char_p[2] = 99
char_p[3] = 100
char_p[4] = 101
char_p[5] = 102
char_p[6] = 103
char_p[7] = 104
char_p[8] = 105
char_p[9] = 106
long_p=[0] = 0
long_p=[1] = 1000000
long_p=[2] = 2000000
long_p=[3] = 3000000
long_p=[4] = 4000000
long_p=[5] = 5000000
long_p=[6] = 6000000
long_p=[7] = 7000000
long_p=[8] = 8000000
long_p=[9] = 9000000
$
```

〔リスト8〕typeofでマクロを作ったソース(test47.c)

```
#include <stdio.h>
#define pointer(T)  typeof(T *)
#define array(T, N) typeof(T [N])
int main(void)
{
    int  ix;
    array (pointer (char), 10) char_p;
    array (pointer (long), 10) long_p;
    for  (ix=0;ix<10;ix++)
    {
         char_p[ix]    =    'a'+ix;
         long_p[ix]    =    ix * 1000000;
    }
    for  (ix=0;ix<10;ix++)
    {
         printf("char_p[%d] = %d¥n",ix,char_p[ix]);
    }
    for  (ix=0;ix<10;ix++)
    {
         printf("long_p=[%d] = %d¥n",ix,long_p[ix]);
    }
    return;
}
```

リスト9のアセンブラソースを見てわかるとおり，ポインタの長さ分，領域が10個ずつ取られているのがわかります．GCCの標準機能で同等の機能を実装することはできません．

● 拡張されたlvalue

複合式，条件式，および，キャストは，そこに含まれる式がlvalueであるかぎり，lvalueとして使うことができます．これは，アドレスを取ったり，値を格納することができるということを意味しています．

複合式の中の最後の式がlvalueであれば，複合式に対して値を代入することができます．以下に示す二つの式は同等です．

```
(a, b) += 5
a, (b += 5)
```

同様に，複合式のアドレスを取ることもできます．次の二つ

の式は同等です.

&(a, b)

a, &b

条件式は，その型がvoidではなく，かつ，真のときに分岐する部分と偽のときに分岐する部分がともに正当なlvalueであれば，正当なlvalueです．たとえば，以下の二つの式は同等です．

(a ? b : c) = 5

(a ? b = 5 : (c = 5))

見てのとおり言語仕様が違うのかと思うくらいに拡張されています．ただこれは，使用してもメリットはないと思います．後で混乱をまねくでしょう．GCCの標準機能を使って一行で同等の機能を実装することはできません．

*　　*

次回は，続けてGNU Cの拡張機能について詳細に説明と検証を行う予定です．

きし・てつお　オフィス岸

〔**リスト9**〕**展開されたアセンブラリスト**(test47.s)

```
    .file     "test47.c"
    .version  "01.01"
gcc2 compiled.:
.section .rodata
.LC0:
    .string   "char p[%d] = %d¥n"
.LC1:
    .string   "long p=[%d] = %d¥n"
.text
    .align 4
.globl main
    .type      main,@function
main:
    pushl %ebp
    movl %esp,%ebp
    subl $96,%esp
    pushl %esi
    pushl %ebx
    nop
    movl $0,-4(%ebp)
    .p2align 4,,7
.L3:
    cmpl $9,-4(%ebp)
    jle .L6
    jmp .L4
    .p2align 4,,7
.L6:
    movl -4(%ebp),%eax
    movl %eax,%edx
    leal 0(,%edx,4),%eax
    leal -44(%ebp),%edx
    movl -4(%ebp),%ecx
    addl $97,%ecx
    movl %ecx,(%eax,%edx)
    movl -4(%ebp),%eax
    movl %eax,%edx
    leal 0(,%edx,4),%eax
    leal -84(%ebp),%edx
    movl -4(%ebp),%ecx
    movl %ecx,%ebx
    movl %ebx,%esi
    sall $5,%esi
    subl %ecx,%esi
    movl %esi,%ebx
    sall $6,%ebx
    subl %esi,%ebx
    sall $3,%ebx
    addl %ecx,%ebx
    movl %ebx,%ecx
    sall $6,%ecx
    movl %ecx,(%eax,%edx)
.L5:
    incl -4(%ebp)
    jmp .L3
    .p2align 4,,7
.L4:
    nop
    movl $0,-4(%ebp)
    .p2align 4,,7
.L7:
    cmpl $9,-4(%ebp)
    jle .L10
    jmp .L8
    .p2align 4,,7
.L10:
    addl $-4,%esp
    movl -4(%ebp),%eax
    movl %eax,%edx
    leal 0(,%edx,4),%eax
    leal -44(%ebp),%edx
    movl (%eax,%edx),%eax
    pushl %eax
    movl -4(%ebp),%eax
    pushl %eax
    pushl $.LC0
    call printf
    addl $16,%esp
.L9:
    incl -4(%ebp)
    jmp .L7
    .p2align 4,,7
.L8:
    nop
    movl $0,-4(%ebp)
    .p2align 4,,7
.L11:
    cmpl $9,-4(%ebp)
    jle .L14
    jmp .L12
    .p2align 4,,7
.L14:
    addl $-4,%esp
    movl -4(%ebp),%eax
    movl %eax,%edx
    leal 0(,%edx,4),%eax
    leal -84(%ebp),%edx
    movl (%eax,%edx),%eax
    pushl %eax
    movl -4(%ebp),%eax
    pushl %eax
    pushl $.LC1
    call printf
    addl $16,%esp
.L13:
    incl -4(%ebp)
    jmp .L11
    .p2align 4,,7
.L12:
    jmp .L2
    .p2align 4,,7
.L2:
    leal -104(%ebp),%esp
    popl %ebx
    popl %esi
    movl %ebp,%esp
    popl %ebp
    ret
.Lfe1:
    .size      main,.Lfe1-main
    .ident    "GCC: (GNU) 2.95.3 20010315 (release)"
```

光ディジタルオーディオ入出力対応

ディジタルオーディオボードの設計/製作

山武一朗

ここでは，先月号の特集「作りながら学ぶコンピュータシステム技術」で掲載できなかった，光ディジタルオーディオ入出力対応のディジタルオーディオボードを設計する．AV機器に使われているディジタルオーディオについて解説した後，ディジタルオーディオデータ転送のシステム負荷を検討し，FIFO内蔵の実用的なオーディオボードを設計する．最後に，WAVEファイル対応の録音/再生サンプルプログラムも作成している．　（編集部）

はじめに

マルチメディアという言葉は，今では死語といわれているようですが（笑），コンピュータで音と映像を扱いたいという要求は，パソコン登場当初からあったようです．

ディジタル回路でもっとも簡単に音を出すとすれば，ロジックの世界の“1”/“0”をそのまま電圧の“H”/“L”に変換し，圧電素子や圧電ブザーを鳴らす，いわゆるビープ音があります．“1”/“0”の切り替え速度を変えることで，音の高低も表現できます．そしてもう少し進化すると，三角波やのこぎり波といったある程度の波形を表現でき，それを3チャネルもった3重和音を出力できるプログラマブルサウンドジェネレータ（PSG）などが登場します．さらに複数の正弦波を重ね合わせたり周波数変調をすることで，より表現力豊かな音を生成できるFM音源なども登場します．

一方で，このように波形を合成して出力するという考え方ではなく，自然界の音をA-D変換してディジタル化し，それを音源として使うサンプリング音源というものも出てきました．技術の進歩とともにA-D/D-A変換のクオリティも向上し，CDオーディオでは16ビット/44.1kHz，DVDオーディオでは24ビット/96kHzなどの量子化数/サンプリング周波数が使われています．

またサンプリングしたデータをそのままファイルに書き出したり，書き出したファイルを出力して，音楽を録音/再生することも可能になりました．

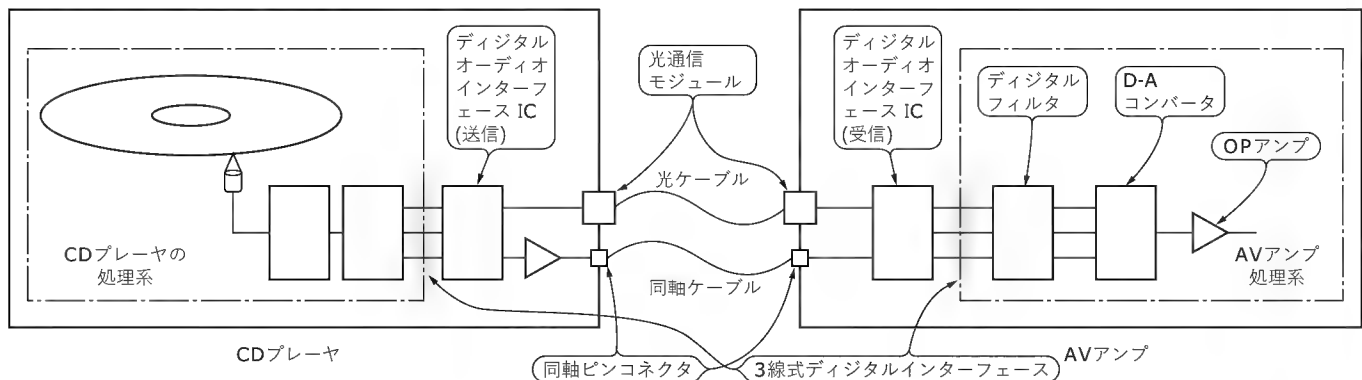
ここでは，CDクオリティで音を入出力できる，光ディジタル対応のオーディオ入出力ボードを設計/製作してみます．

ディジタルオーディオの基礎

● ディジタルオーディオ機器の構成例

図1にディジタルオーディオシステムの構成例として，ディジタルオーディオ出力対応のCDプレーヤと，ディジタルオーディオ入力対応のAVアンプの，ディジタルオーディオ信号の流れを示します．

現在の低価格化が進み内部がシステムLSI化されたAV機器は，かなりの部分が1チップ化されているでしょうが，基本的な考え方は通じるものがあると思います．また**図1**ではCDプレーヤとAVアンプという別々の機器ですが，アナログ出力をもっているディジタルオーディオ機器であれば，その内部は**図1**のシステムが一つの機器に内蔵されていると考えることができます．

〔図1〕ディジタルオーディオシステムの構成例

● 同軸/光ディジタル端子

一般的な民生用AV機器に使われるディジタルオーディオインターフェースには，端子がピンプラグ形状の同軸インターフェースと，光ファイバを使って伝送する光インターフェースの2種類が存在します(**写真1**)．筆者の感覚では，光ディジタルインターフェースのほうがよく使われているように思えます．高級機には同軸もよく使われています．どちらも一長一短があるようで，同軸がなくなるというものでもないようです．

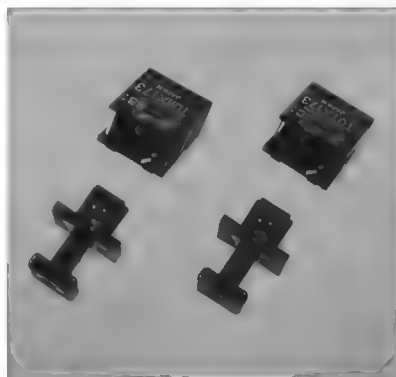
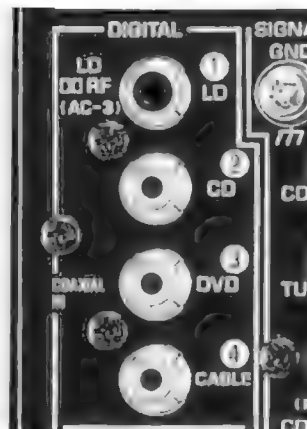
これらは信号を伝送するための物理的な媒体が異なるだけで，その上を流れる信号は同じものと考えて間違いありません．光ディジタルオーディオ信号は，光ディジタルオーディオレシーバ/トランシーバモジュールで，同軸ディジタルオーディオと同等の信号に変換されます．ちなみに同軸ディジタルオーディオ信号も，実質的には信号のドライブ能力などの問題で，バッファICを経由して外とつながっていることが多いようです．

● ディジタルオーディオインターフェースIC

ディジタルオーディオの伝送には，同軸にしろ光にしろ，1本のケーブルだけで行われます．この1本の信号でL/Rの両チャネルのディジタルオーディオデータはもちろん，サンプリング周波数の情報やそのほかの付随情報などもいっしょに伝送しています．

よって出力(送信)側のディジタルオーディオインターフェースICは，ディジタルオーディオ信号やクロック情報，その他の情報を合成して1本の信号にまとめて出力します．逆に入力(受信)側のディジタルオーディオインターフェースICは，送信側のクロックに同期をかけ，1本の信号からディジタルオーディオ信号やクロック情報，その他の情報を取り出します．

ここで**図1**のAVアンプ側，受信側のディジタルオーディオインターフェースICとD-Aコンバータの間に注目してください．ここで伝送されている情報として，ビットクロック，L/Rクロック，そしてオーディオデータの3種類の信号があります．これはディジタルオーディオ制御系のICでよく使われる信号で，3線式ディジタルインターフェースと呼ばれることがあります．

〔写真1〕ディジタルオーディオインターフェースの端子の外観

(**a**) 光ディジタル端子
(左：受信用，右：送信用，下は端子保護キャップ)

(**b**) 同軸ディジタル端子

ここを切り口にすることで，メーカーの異なるインターフェースICとディジタルフィルタやD-Aコンバータを接続できるのです．

● 3線式ディジタルインターフェース

図2に3線式ディジタルインターフェースの信号波形例を示します．量子化数は16ビットが一般的なので，ビットクロックが16クロックごとに，L/Rクロックの極性も切り替わります．しかし中には将来の拡張性をにらんでか，ビット数が20ビットや24ビットに対応したものもあります．その場合，16ビットのディジタルオーディオを伝送する場合，データをLSB側に詰めるかMSB側に詰めるかなど，使用するインターフェースICやディジタルフィルタなどにより設定が必要になります．

オーディオデータはシリアルなので，量子化したディジタルオーディオデータをMSB側から出力するのが一般的のようです．またL/RチャネルはLチャネルを先に伝送します．

なお，ディジタルオーディオインターフェースICの動作用システムクロック周波数としては，サンプリング周波数の256倍(256fs)や384倍(384fs)という周波数が使われます．3線式ディジタルインターフェースの切り口で別のメーカーのICをそれぞれ組み合わせることができますが，システムクロックは同期が取れている必要があるので，たとえば256倍のクロックを採用する場合は256fs対応のデバイスでそろえる必要があります．

なお筆者は，DVD時代のディジタルオーディオ設計には詳しくないので，現在ではより高い量子化数/サンプリング周波数に対応する新しい内部インターフェースが一般的になっているかもしれません．

2 ディジタルオーディオボードの仕様検討

● 対応ディジタルオーディオの仕様

DVDでは5.1チャネルなどのサラウンド信号も，ディジタルオーディオインターフェースを経由して伝送します．しかしドルビーディジタルなどのサラウンド信号はいろいろな特許が絡んでいるので，残念ながらおいそれと使うわけにはいきません．

そこで今回は，ディジタルオーディオとしてもっとも一般的な，44.1kHz，48kHz，32kHzの三つのサンプリング周波数への対応を考えます．量子化数は16ビットで，L/Rのステレオという仕様は固定です．

● ディジタルオーディオインターフェースIC

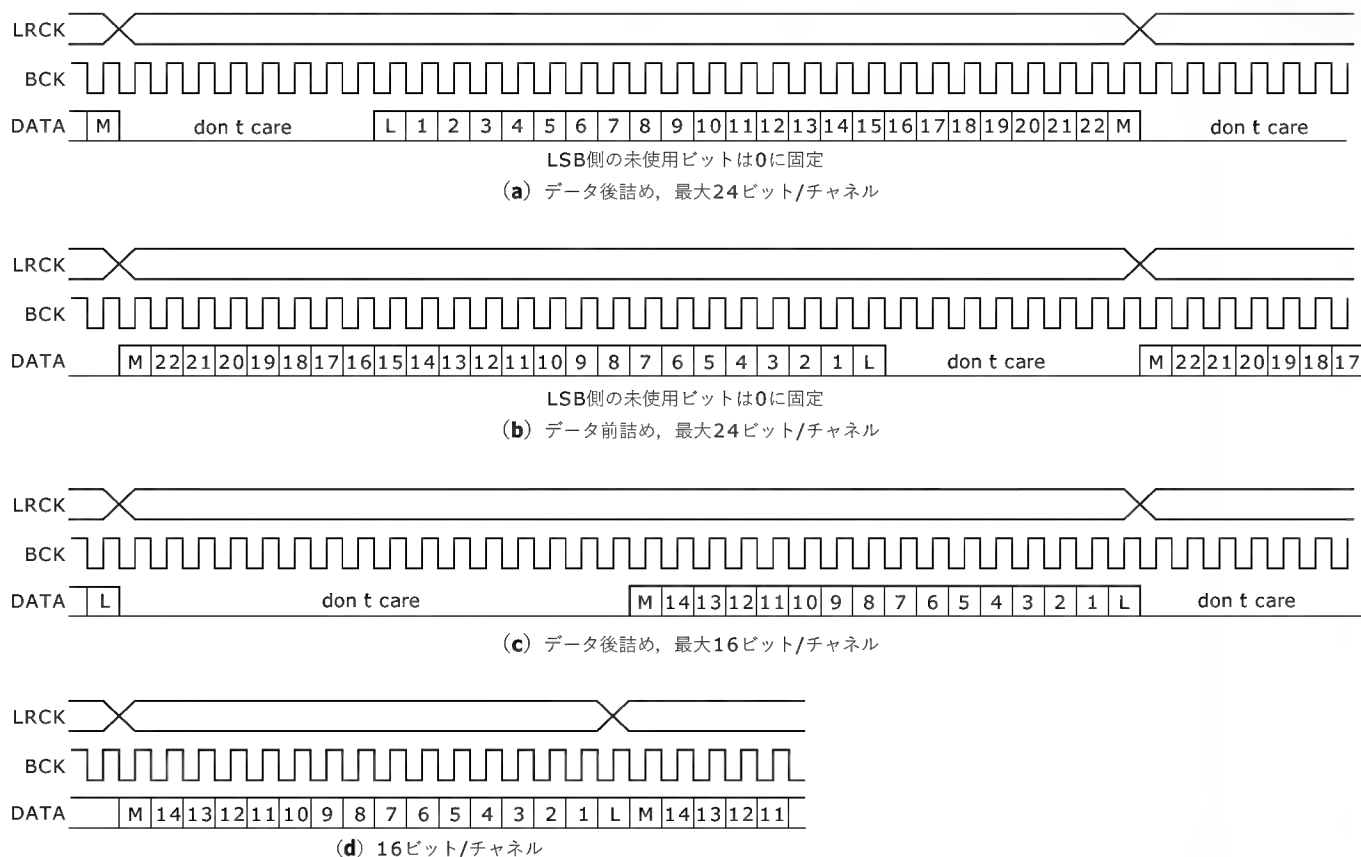
すでに説明したようにディジタルオーディオインターフェースでは，一つのケーブルでクロックとデータがいっしょに送られてきます．送信側ではクロックとデータを合成する回路が，受信側では送信側のクロックに同期をかけてデータとクロックを分離する回路が必要になります．この部分にはアナログ回路が必要になるので，通常のFPGAだけでは実現できません．

〔図2〕3線式ディジタルインターフェースの波形例

(**a**) データ後詰め，最大24ビット/チャネル

(**b**) データ前詰め，最大24ビット/チャネル

(**c**) データ後詰め，最大16ビット/チャネル

(**d**) 16ビット/チャネル

そこで今回は，3種類のサンプリング周波数に対応した，ディジタルオーディオ信号を送受信する専用ICを使い，それらのロジック出力部分をPCIバスにブリッジし，制御する部分をFPGAで実現するという方法を採ります．

今回採用したディジタルオーディオインターフェースICは，送信用にはTC9271F(東芝)を，受信用にはTC9245N(同)を使いました．パッケージはどちらも同じ28ピンSOPです．**表1**にピン配置を示します．

送信側のTC9271Fには，3線式ディジタルインターフェースの仕様にいくつかのモードがあります．今回はMSBファースト，16ビット/チャネル，データ後詰めの設定で動作させることにします．また送信側には，送信するディジタルオーディオ信号のサンプリング周波数の設定や，エンファシスの設定ピン，そして動作クロック周波数の入力ピンなどもあります．

受信側のTC9245Nにはいくつかの動作モードがあります．外付けにマイコンを置き，それによって制御させる場合にはシリアルモードを使いますが，今回は後述するようにFPGAと接続することを考えているので，パラレルモードを使いました．また，現在受信しているディジタルオーディオ信号のサンプリング周波数やエンファシスの状態，コピー禁止信号が立っているかどうかのステータス情報も出力されます．なお，正常な信号が入力されていない状態では，エラー信号が出力されます．

● 光インターフェースの採用

また，同軸と光インターフェースの2種類が存在しますが，今回は光インターフェースを採用することにします．その理由としては，本ボードはPCIバス対応のボード，つまりノイズの巣窟のようなパソコンに実装することになるので，そのノイズを外のAV機器に伝えないためにも，機器間が絶縁される光インターフェースが望ましいと考えたためです．

以上をまとめたディジタルオーディオボードのブロック図を**図3**に示します．

● TC9271FとTC9245Nの仕様

受信側TC9245Nは16ビットMSBファースト，Lチャネル先出しという仕様で固定されています．**図2**でいうところの(**d**)の波形になります．送信側TC9271FはICのピンを設定することで，いくつかのモードを選択できます．ここでは受信側と同じ仕様にしています．こうすることで，ディジタルオーディオ入力をそのままスルーして出力端子から出力する場合に便利です．

● クロックセレクタ

ディジタルオーディオインターフェースICは，受信側は受信したディジタルオーディオ信号に同期をかけてクロックを生成するので，外付けのクロックは不要です．FPGAにはこれを入力して受信側ブロックを動作させます．

しかし，送信側はどのサンプリング周波数で出力するか，サ

〔表1〕ディジタルオーディオインターフェースICのピン配置

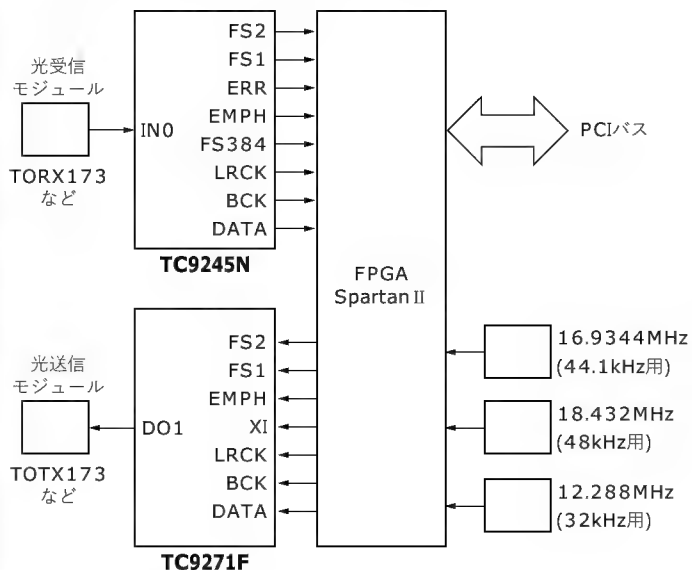
端子番号	記　号	I/O	機能説明
1	BLOCK	O	ブロック先頭位置出力端子
2	UBDA	I	ユーザービット入力端子
3	LRS	I	LRCK極性選択端子 LRS \ LRCK“L” / LRCK“H” “L”レベル：Rチャネルデータ / Lチャネルデータ “H”レベル：Lチャネルデータ / Rチャネルデータ
4	LRCK	I	L/Rクロック入力端子
5	BCK	I	ビットクロック入力端子
6	DATA	I	ディジタルオーディオデータ入力端子
7	VLDY	I	補正フラグ入力端子
8	EMPH	I	エンファシスフラグ設定端子
9	COPY	I	コピーフラグ設定端子
10	FS1	I	サンプリング周波数設定端子1
11	FS2	I	サンプリング周波数設定端子2
12	CKS	I	クロック分周波数選択端子
13	XI	I	クロック入力端子
14	V_{SS}	—	グラウンド端子
15	DO1	O	ディジタルデータ出力端子1
16	DO2	O	ディジタルデータ出力端子2
17	M1	I	チャネルモード設定端子1
18	M2	I	チャネルモード設定端子2
19	IS1	I	データ入力モード設定端子1
20	IS2	I	データ入力モード設定端子2
21	CGT1	I	カテゴリコード設定端子1
22	CGT2	I	カテゴリコード設定端子2
23	CGT3	I	カテゴリコード設定端子3
24	FR32	O	FR32出力端子
25	LBIT	I	LBIT入力端子
26	CKA1	I	クロック精度設定端子
27	CKA2	I	クロック精度設定端子
28	V_{DD}	—	電源電圧端子

(a) 送信側TC9271F(パラレルモード/2チャネル)

端子番号	記　号	I/O	機能説明
1	DOUT	O	IN0～IN2選択出力端子． IN3選択時は“L”固定出力
2	SEL	I	“H”またはオープンでパラレルモード，“L”でシリアルモードに対応
3	IN0	I	ディジタルオーディオデータ入力端子 (チャネル0～4)
4	IN1	I	
5	IN2	I	
6	IN3	I	
7	FCONT	O	PLLミスロック検出信号出力端子
8	PD	O	位相比較器位相誤差信号出力端子
9	V_{DDA}		アナログ電源電圧端子
10	AMPI	I	LPF用OPアンプ入力端子
11	VCOI	O	OPアンプ出力端子(VCO発振制御電源出力)
12	VCOINH	I	VCO発振停止制御用入力端子
13	NC	—	無接続端子
14	V_{SSA}	—	アナロググラウンド端子
15	V_{SS}	—	ディジタルグラウンド端子
16	DATA	O	ディジタルオーディオデータ出力端子
17	BCK	O	ビットクロック出力端子(32fs)
18	LRCK	O	L/Rクロック出力端子(L/Rチャネル極性固定)
19	EMPH	O	エンファシス出力端子(“H”でエンファシスあり)
20	$\overline{\text{ERROR}}$	O	エラー検出フラグ出力端子(“L”でエラー検出)
21	$\overline{\text{TEST}}$	I	テスト用入力端子
22	FS384	O	384fsクロック出力端子
23	S1	I	入力端子
24	S2	I	入力端子
25	$\overline{\text{COPY}}$	O	コピー禁止フラグ出力
26	FS1	O	サンプリング周波数デコード出力端子1
27	FS2	O	サンプリング周波数デコード出力端子2
28	V_{DD}	—	ディジタル電源電圧端子

(b) 受信側TC9245N(パラレルモード)

〔図3〕ディジタルオーディオボードのブロック図

光受信モジュール
TORX173など
IN0
FS2
FS1
ERR
EMPH
FS384
LRCK
BCK
DATA
TC9245N
FPGA
Spartan II
PCIバス
光送信モジュール
TOTX173など
DO1
FS2
FS1
EMPH
XI
LRCK
BCK
DATA
TC9271F
16.9344MHz (44.1kHz用)
18.432MHz (48kHz用)
12.288MHz (32kHz用)

ンプリング周波数設定端子に設定するだけでなく，実際にサンプリングの何倍かのクロックが必要です．TC9271Fは384fs系にも256fs系にも自動的に対応するので，今回は外部に386倍のクロックを実装します．

さらに今回は3種類のサンプリング周波数に対応するので，44.1kHz用に16.9344MHzを，48kHz用に18.432MHzを，32kHz用に12.288MHzの三つのクロックを実装し，FPGAに実装しています．FPGAの内部でサンプリング周波数を設定すると，その設定値がTC9271Fのサンプリング周波数設定端子に出力すると同時に，FPGA内部のクロックセレクタで該当クロックを選択し，TC9271Fのクロック入力に出力されるようにします．

● PCIバスへのブリッジ

すでに説明したように，今回採用したディジタルオーディオインターフェースは，ビットクロック，L/Rクロック，そしてビットデータからなる3線式ディジタルインターフェースを使っています．つまりディジタルオーディオ信号をシリアルで伝送し

ているわけです．

しかし，ディジタルオーディオ信号をパソコン上で扱うには，パラレルに変換してメモリやストレージに格納します．今回の仕様ではL/Rステレオの16ビットですから，ちょうど32ビットにL/Rの1サンプリング分を格納すると都合がよいと考えられます．

● もっとも単純なI/Oデバイス

L/Rまとめて1サンプリングがちょうど32ビットということは，PCIバスの幅とちょうど一致し，PCIバス上で1バスサイクルで1サンプリングのデータの入出力が可能になります．これをもっとも単純に読み書きするには，ステータスレジスタを設けて，データ転送の準備が整ったというステータスビットが立ったら，データレジスタを32ビット幅で読み書きするというポーリング制御でいけそうです．

● 転送負荷の検討

このとき，どれくらいの負荷がかかるのかを計算してみます．今回は量子化数16ビットでステレオ2チャネルで，これを32ビット幅で1度に転送します．これは今回の仕様では一定なので，サンプリング周波数によりそのデータ転送転送の頻度が変わることになります．もっとも一般的なディジタルオーディオ信号と思われるCDオーディオを考えてみると，そのサンプリング周波数は44.1kHz，つまり22μsに1回のデータ転送が発生するわけです．

22μsというと，それほど速くないように思われます．たとえば最近のマイコンではRISC的に1命令1クロック動作のものが増えていますが，かりに10MHzで動作していたとしても1クロック100nsなので，220命令は実行できる計算になります．もちろん命令フェッチだけでなくデータの読み書きやLOOP命令や条件分岐命令の実行も必要なので，実質はもっと余裕がなくなるでしょう．そしてもう一つ見逃してはならない点は，常に22μsに1回という定期的なタイミングを維持しつつ，データ転送を行わなければならないという点です．少しでも間に合わなければノイズや音切れなどが発生し，オーディオシステムとしては欠陥となります．

ここで，少し長めの割り込み処理が発生したとすればどうでしょうか．割り込み処理ルーチンの命令数的には数十ステップでも，一般的な割り込み処理ではレジスタの退避/復帰処理などが発生するので，あっという間に数μsオーダのポーリング不能時間が発生するでしょう．

● 割り込み処理のオーバヘッド

また，ステータスビットが立った瞬間に割り込みを出力し，割り込みルーチン内でデータ転送を行う方法ではどうでしょうか．これも，もしほかの割り込み処理の中で，22μs以上に時間がかかる処理があった場合は間に合わなくなります．ただし，多重割り込みシステムとして，ディジタルオーディオ転送を最上位の割り込みレベルで処理すれば，安定した割り込み処理を実現できるかもしれません．しかしこの方法でも，割り込み処理にはレジスタの退避/復帰処理などが発生するので，22μsごとにデータ転送を行っていたのでは，その前後に毎回レジスタの退避/復帰処理が発生することを意味します．限られたシステム処理能力をむだに使うことになります．

● 送受信独立して256ワードのFIFOを実装

このように毎回22μsごとに……という処理は，決して軽い処理ではありません．また1ワードごとに毎回割り込みを発生させることもまた，かなりの負荷になります．

そこで，1度の割り込みでまとめてデータ転送を行うと，システムにおけるデータ転送の負荷を大幅に減らすことができます．

そこで，ディジタルオーディオインターフェースICとPCIバスの間にFIFOバッファを設け，ここである程度の量になるまでデータをバッファリングし，一定量に達したら一気にデータ転送を行うという方法を採用します．

● FIFOと割り込み

今回はデータの入出力部分にFIFOを実装したので，FIFOの状態によりステータスの変化や割り込みを出力するようにします．問題は，FIFOをどのように制御するかです．

一気に転送するといっても，ブロック単位のデータ転送で，ブロックとブロックの間にACKを返すといったようなプロトコルの場合は，ブロックサイズでFIFOを確保し，FIFOフルでFIFOを一気に読み出し，FIFOエンプティで一気にブロックサイズ分だけ書き込みを行うなどの方法を使います．

しかし今回は，常時転送が行われるというシステムです．フルやエンプティ状態になってからやおら作業を始めては，FIFOがあふれる，もしくは空になってしまいます．

そこで受信FIFOはFIFOの空きが半分以下になったら，つまりハーフフルで入力データレディ状態を返し，送信FIFOは半分以上空きがあれば出力データエンプティ状態を返します．またそれぞれの要因で，システムに割り込みを出力するようにします．

● PCIデバイス仕様

表2(a)にPCIデバイスのコンフィグレーションレジスタ仕様を示します．ベースアドレスレジスタは1本のみを使い，I/O空間を要求しています．要求空間サイズは，今回は32バイトとしています．

また，**表2(b)**にI/Oレジスタ仕様を示します．大きく分けて，割り込み制御のためのレジスタと，FIFO制御のレジスタ，そしてディジタルオーディオインターフェースICの制御ピンレジスタの状態を設定/ステータスレジスタを割り当てています．

3 ディジタルオーディオボードの設計/製作

● VHDLによるFPGAの設計

今回は前回の特集でも使用したPCI試作評価用ボードを使い，拡張用ピンヘッダにユニバーサル基板をスタック接続して，TC9271FやTC9245N，そして送信用クロックを実装しました．**写**

〔表2〕PCIデバイス仕様

ベンダID	0x6809
デバイスID	0x8007
クラスコード	0x04 0x01 0x00 マルチメディア/オーディオ
ヘッダタイプ	0x00
ベースアドレス0	I/O空間 32バイトの空間を使用
割り込み	INTA#使用

(a) コンフィグレーションレジスタ仕様

オフセット	レジスタビット	名称
+00h	32	割り込みステータスレジスタ ビット1　ディジタルオーディオ出力データエンプティ ビット0　ディジタルオーディオ入力データレディ
+04h	32	割り込みマスクレジスタ ビット1　ディジタルオーディオ出力データエンプティマスク ビット0　ディジタルオーディオ入力データレディマスク
+10h	32	ディジタルオーディオFIFOリセット制御 ビット4　ディジタルオーディオ出力FIFOリセット ビット1　ディジタルオーディオスルー出力設定ビット ビット0　ディジタルオーディオ入力FIFOリセット
+14h	32	ディジタルオーディオ入力出力ステータス/制御レジスタ ビット31　ディジタルオーディオ出力アンダーランエラー ビット30　ディジタルオーディオ出力データエンプティ ビット29　ディジタルオーディオ出力バッファフル ビット19　ディジタルオーディオ出力エンファシス設定 ビット17　ディジタルオーディオ出力サンプリング周波数設定1 ビット16　ディジタルオーディオ出力サンプリング周波数設定0 ビット15　ディジタルオーディオ入力オーバランエラー ビット14　ディジタルオーディオ入力データレディ ビット13　ディジタルオーディオ入力バッファエンプティ ビット4　ディジタルオーディオ入力エラーステータス ビット3　ディジタルオーディオ入力エンファシスステータス ビット1　ディジタルオーディオ入力サンプリング周波数ステータス1 ビット0　ディジタルオーディオ入力サンプリング周波数ステータス1
+18h	32	ディジタルオーディオ入力データ(読み出し時) ディジタルオーディオ出力データ(書き込み時)

(b) I/Oレジスタ仕様

真2に試作したディジタルオーディオボードを示します.

なお，ディジタルオーディオインターフェースICのピン配置は表1に，ブロック図は図3に示しているので，回路図は省略します．FPGA側はいずれにI/Oピンでも問題ありません.

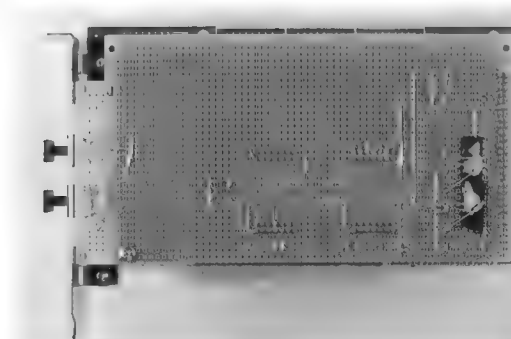
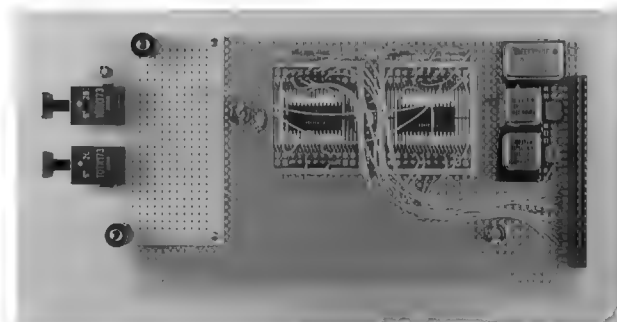
また，2003年1月号の特集同様，FPGAの設計にはVHDLを使っています．リスト1(p.155)に設計したVHDLのソースの一部を示します．このVHDLソースは，参考文献2)のPIO.VHDをベースにしています．PCIバス側のシーケンサの動作などは，そちらを参照してください.

● FIFOモジュール

今回使用したPCI試作評価用ボードには，Spartan IIが実装されています．デバイスメーカーのザイリンクスは，同社のFPGAデバイスを活用するための，さまざまな回路モジュールをIPコアとして使えるように公開しています．これらはLogiCOREと呼ぶ，いくつかのパラメータをGUI上で設定して回路モジュールを生成する機能を使って簡単に活用できます.

しかし残念なことに，ユーザー登録すれば誰でもWebからダウンロードして使えるフリー版の設計ツールISE WebPACKでは，このLogiCOREの機能が使えません.

ただし，LogiCORE機能の使える正式版の設計ツールISE FoundationやISE Allianceで回路モジュールを生成すれば，そのモジュールに手を加えることはできませんが，そのまま上位階層からその回路モジュールを呼び出して使うことは可能です．FIFOモジュールのビット幅や深さなどを変更しなければ，それ以外の部分はISE WebPACKでも設計変更して再論理合成，再配置配線してみることができます.

〔写真2〕試作したディジタルオーディオボードの外観

(a) PCI試作評価ボードに実装したようす

(b) スタック接続基板の裏側

● 幅32ビット/深さ256ワードのFIFO

2003年1月号の特集の趣旨を尊重するには，FIFOの深さの違いによるシステムパフォーマンスの変化を手元でテストできるように，自由にFIFOの深さを変えられるようにするべきですが，ISE WebPACKではそれができないため，正式版設計ツールで深さ256ワードのFIFOモジュールを用意し，それを使うことにします．FIFOの深さや幅などのパラメータは変更できません．

● シリアル→パラレル変換部

ディジタルオーディオ入力部分は，シリアルデータをパラレルに変換し，入力FIFOに格納する部分です．入力のオーディオデータはビットクロックの立ち上がりエッジでサンプリングするので，ビットクロックの立ち上がりに同期して動作させます．

先にLチャネルが送信されてくるわけですが，16ビット分シフトレジスタに格納したからといって，すぐに入力FIFOに書き込みはできません．今回使用したFIFOモジュールは32ビット幅なので，L/R両チャネルのデータがそろった時点で，32ビットを一度に書き込みする必要があるからです．

そこでL/Rクロックが"H"になったクロックで，その時点でのシフトレジスタに残っているデータを書き込むように記述しています．また，このとき入力FIFOがフル状態であれば，ディジタルオーディオ入力オーバランエラーのフラグを立てます．

また，シフトレジスタ処理は，とくに16ビット分をカウントすることなく，ビットクロックの立ち上がりエッジが入力されれば常に1ビットシフトするように設計しています．今回は3線式ディジタルインターフェース部分で，ビット長を16ビットにしているためで，オーバフローのような処理は，考慮する必要はありません．

● パラレル→シリアル変換部

ディジタルオーディオ出力部は，出力FIFOからデータを読み出して，パラレルデータをシリアルに変換して出力する部分です．

入力部はTC9245Nから出力されるビットクロックに同期すればよいわけですが，出力部はFPGA側からビットクロックを出力しなければなりません．外付けのクロックは，サンプリング周波数を384倍したクロックが実装されているので，これをまず1/6分周し，さらに1/12分周してビットクロックを生成します．ただし，直接カウンタで12までカウントしてリセットをかける方法では，デューティ50%のクロックを生成しづらいので，1/6分周のカウンタを作ってその中でビットクロック信号をトグル動作させることで，1/12分周したデューティ50%のビットクロックを生成します．また，それを16ビット分カウントして，L/Rクロックも生成します．

パラレル→シリアル変換部では，ビットクロックで32ビットのカウントがクリアされる2クロック前で出力FIFOの読み出しを開始し，次のクロックでデータを取り出し，カウント0の状態からシフト出力動作を行います．最初はLチャネルのMSBからです．なお，出力FIFOからデータを読み出しするとき，FIFOがエンプティ状態だった場合は，ディジタルオーディオ出力アンダーランエラーのフラグを立てます．

● FIFO制御

仕様の検討で説明したように，今回はハーフフルの状態でシステムに割り込みを出力します．今回はLogiCOREで生成したFIFOモジュールの仕様から，入力FIFOバッファでは書き込み側のFIFOライトポインタの，出力FIFOでは読み出し側のFIFOリードポインタの，それぞれ最上位ビットを使ってハーフフル/エンプティ状態として処理しています．

4 制御ソフトウェアの作成

● ストレージが必須

ディジタルオーディオのデータ量は，CDクオリティで1曲5分とすると，約50Mバイトものデータ量になります．そのままメモリにおいておける量ではありません．HDDやCD-ROMといったストレージデバイスへの読み書きが必須になります．

2003年1月号で設計したSH-4によるオリジナルコンピュータシステムでは，まだOSと呼べるものは動作していません．HDDへの読み書きは，テストプログラムに毛が生えた程度の，セクタダンププログラムが動いている程度で，ファイルシステムがありません．ファイルフォーマットを考えず，セクタ単位でHDDへベタ書きという手もありますが，それはあまりに乱暴です(専用HDDオーディオレコーダというのもそれはそれでおもしろいが)．

そこで今回は，動作確認用サンプルプログラムとして，PC/ATのMS-DOS環境で動作する，録音再生ツールを作成してみました．

● WAVEファイルの録音/再生ツール

さて，録音する場合はディジタルオーディオ入力から入力したデータをそのままバイナリにしてファイル化し，逆に再生する場合はそのバイナリファイルを読み込んで出力しても，システムとしては一貫したものが成り立つでしょう．2003年1月号特集の第4章のグラフィックスカードでは，プロローグで説明のあるようにBMPファイルのローダを作成しました(ただし，こちらもファイルシステムは存在しないので，BMPファイルはメモリ上に確保したバッファ領域から取り出す構造になっている)．どうせならこちらも，何か一般的なファイルフォーマットに対応させたいものです．

AT環境で一般的なオーディオファイルといえば，現在ではWAVEファイルが一般的でしょう．そこで，ディジタルオーディオ入力からの入力をWAVEファイルとして出力(録音)し，逆にWAVEファイルをディジタルオーディオ出力から出力する(再生)するツールを作成しました．

● WAVEファイルのフォーマット

Windows標準のWAVEファイル(拡張子"WAV")は，RIFFというフォーマットの一種です．**表3**にWAVEファイルのフォーマットを示します．基本的にはヘッダがあり，それ以降にデ

〔表3〕WAVEファイルのフォーマット

バイト数	名　称
4バイト	RIFF形式の識別子‘RIFF’
4バイト	ファイルサイズ(バイト単位)
4バイト	RIFFの種類を表す識別子‘WAVE’
4バイト	タグ1参照
4バイト	データの長さ1
nバイト	データ1
4バイト	タグ2参照
4バイト	データの長さ2
nバイト	データ2
(以下同様)	

(a) RIFFフォーマット全体の構造

バイト数	名　称
4バイト	“fmt ”フォーマット定義．最初のタグ(必須) (4文字目の' '(スペース)も含まれるので注意)
4バイト	“fact”全サンプル数(波形データの前に存在する)
4バイト	“data”波形データ(必須)

(b) WAVEで使われるおもなタグ

バイト数	名　称
4バイト	“data”dataフォーマットの定義
4バイト	バイト数　dataのバイト数
nバイト	実際のデータ ● ステレオならばＬＲＬＲＬＲ…の順番 ● 数値の並べ型はインテルバイトオーダ ● ビット数は8ビットまたは16ビット ● 8ビットならば符号なしunsigned(0～255，無音は128) ● 16ビットならば符号付きsigned (−32768～＋32767，無音は0)

(d) データタグのフォーマット

バイト数	名　称
4バイト	“fmt ” fmtフォーマットの定義
4バイト	バイト数　fmtのバイト数 リニアPCMならば16(10h 00h 00h 00h)
2バイト	フォーマットID参照　リニアPCMならば1(01h 00h)
2バイト	チャネル数　モノラルならば1(01h 00h)，ステレオならば2(02h 00h)
4バイト	サンプリングレート　44.1kHzならば，44100(44h ACh 00h 00h)
4バイト	データ速度(バイト/秒)　44.1kHz/16ビットステレオならば，44100 × 2 × 2 = 176400(10h B1h 02h 00h)
2バイト	ブロックサイズ(バイト/サンプル×チャネル数)　16ビットステレオならば，2 × 2 = 4(04h 00h)
2バイト	サンプルあたりのビット数(ビット/サンプル)．WAVEフォーマットでは8ビットか16ビット．16ビットならば10h 00h)
2バイト	拡張部分のサイズ(リニアPCMならば存在しない)
nバイト	拡張部分(リニアPCMならば存在しない)

(c) fmtタグのフォーマット

ータが続きます．

本ボードの仕様的としては16ビット/ステレオが基本ですが，WAVEファイルでは8ビットやモノラルというファイルも存在します．そこで再生するファイルがモノラルだった場合は同じデータをL/R両チャネルに出力し，8ビットだった場合は次式，

16ビットデータ＝(8ビットデータ－128)<<8

で16ビットに変換します(8ビット時は0～255，無音が128，16ビット時は－32768～32767，無音が0)．また，周波数が22kHzや11kHzの場合は，同じサンプリングデータを2回または4回，出力FIFOに書き込むことで，44kHzサンプル分のデータにして44.1kHzの周波数で出力するようにしています．

録音は，本ボードが対応している3種類のサンプリング周波数のみで，入力されたディジタルオーディオのサンプリング周波数を自動判別してヘッダに書き出します．量子化数は16ビットでステレオという仕様は固定です．

● プログラムの動作

リスト2(p.156)に，作成したWAVEファイルの録音/再生ツールを示します．MS-DOS環境で動作するので，いつものように参考文献2)のPCIデバッグライブラリ for DOSを使って，16ビットMS-DOS環境で，32ビットI/Oアクセスや割り込み処理を行っています．

まず**表2(a)**のコンフィグレーションレジスタの仕様で示すベンダIDやデバイスIDを検索し，ベースアドレスレジスタ0と割り込みラインレジスタを読み出します．正しくリソースが割り当てられているようであれば，そのI/Oアドレスをベースに，割り込みレジスタや割り込みエントリを初期化します．

まとめ

今回は，L/Rステレオが1チャネルの入力と出力ができるのみという，サウンドカードとしてもっとも基本的な仕様のみでした．より本格的に活用するには，左右のバランス調整や，全体の音量調整機能も必要でしょう．また，ステレオ1チャネルだけでなく複数チャネルを実装し，ハードウェア的なミキサ機能を実現することで，より表現力豊かなサウンドカードを実現できるでしょう．

参考文献

1) 特集「作りながら学ぶコンピュータシステム技術」，『Interface』，2003年1月号
2)『PCIデバイス設計入門』，TECH I Vol.3，CQ出版(株)

やまたけ・いちろう　来栖川電工有限会社

〔リスト1〕**PCIデバイスのVHDLソース**（一部）

```
                    ～中略～
-- DAI入力データレディ/DAO出力データエンプティフラグ制御(PCIバスクロック同期処理)
-- DAI入力オーバーラン/DAO出力アンダーランエラーフラグ制御(PCIバスクロック同期処理)

DAxtoPCI Equ : process(PCICLK, PCIRST n)
begin
    if (PCIRST n = '0') then
         DAI DataRdy  <= '0';
         DAI OverRun  <= '0';
         DAO DataEmp  <= '1';
         DAO UnderRun <= '0';
    elsif (PCICLK'event and PCICLK = '1') then
         DAI OverRun  <= DAI OverRun f;
         DAI DataRdy  <= DAI wr count(1);
         DAO UnderRun <= DAO UnderRun f;
         DAO DataEmp  <= not (DAO rd count(1) or DAO rd count(0));
    end if;
end process DAxtoPCI Equ;


-- ディジタルオーディオ入力シフトレジスタ処理
-- DAI BCKクロック入力
iDAI BCK PAD : IBUF
    port map (
         I    => DAI BCK ,
         O    => iDAI BCK
    ) ;

DAI wr clk <= iDAI BCK;      -- DAI入力FIFO書き込みクロック

DAI DATA Equ : process(iDAI BCK, PCIRST n, DAI FIFO Reset)
    variable DAI LRCKf : std logic;
begin
    if (PCIRST n = '0') or (DAI FIFO Reset = '1') then                -- PCIバスリセットがアサートされたとき
         DAI ainit     <= '1';    -- DAI入力FIFOリセット
         DAI wr en     <= '0';
         DAI OverRun f <= '0';
         DAI ShiftRegL <= (others => '0');
         DAI ShiftRegR <= (others => '0');
         DAI LRCKf     := '0';
    elsif (iDAI BCK'event and iDAI BCK = '1') then
         DAI ainit     <= '0';    -- DAI入力FIFOリセット解除

         if (DAI LRCKf = '0') and (DAI LRCK = '1') then               -- DAI LRCK立ち上がり(Lch開始)
              if (DAI full = '1') then
                   DAI OverRun f <= '1';
              else
                   DAI din(31 downto 16) <= DAI ShiftRegR;            -- DAI入力FIFO書き込み
                   DAI din(15 downto  0) <= DAI ShiftRegL;
                   DAI wr en <= '1';
              end if;
         else
              DAI wr en <= '0';
         end if;
         if (DAI LRCK = '1') then-- Lchデータ受信時
              DAI ShiftRegL(0) <= DAI DATA;                           -- シフトレジスタ
              DAI ShiftRegL(15 downto 1) <= DAI ShiftRegL(14 downto 0);    -- MSBファースト
         else                        -- Rchデータ受信時
              DAI ShiftRegR(0) <= DAI DATA;                           -- シフトレジスタ
              DAI ShiftRegR(15 downto 1) <= DAI ShiftRegR(14 downto 0);    -- MSBファースト
         end if;
         -- ↑エッジを検出するまで、ひたすらシフトして受信していく
         DAI LRCKf := DAI LRCK;
    end if;
end process DAI DATA Equ;

-- 384fsクロック選択
iDAO XI   <= OSC 12M when (DAO FS1 Reg = '1') and (DAO FS2 Reg = '1') else          -- 32kHzモード
              OSC 16M when (DAO FS1 Reg = '0') and (DAO FS2 Reg = '0') else          -- 44.1kHzモード
              OSC 18M when (DAO FS1 Reg = '0') and (DAO FS2 Reg = '1') else          -- 48kHzモード
              DAI FSEXT;      -- DAIクロック選択

-- BCKクロック分周
DAO BCK Equ : process(iDAO XI)
begin
    if (iDAO XI'event and iDAO XI = '1') then
         if (CLK CountA = "101") then             -- 6回目(5)カウント
              CLK CountA <= (others => '0');
              DAO BCKout <= not DAO BCKout;      -- ビットクロックトグル動作(1/12分周)
         else
              CLK CountA <= CLK CountA + 1;
                   ～以下略～
```

〔リスト2〕WAVEファイルの録音/再生ツールのプログラム(-部)

```
                    ～中略～
/* ディジタルオーディオ入出力割り込み処理 */
void PCI Int waveOut(void)
{
  int i;
  unsigned char c;
  unsigned long l;
  c = inp(IO Address+0);                  /* ステータス確認 */
  /* DAO出力データエンプティ割り込み */
  for(i=0;i<128;i++) {                    /* 128ワード読み出し */
    l = *INT ReadBufferPtr++;             /* 読み出しバッファ */
     IoWriteLong(IO Address+0x18, l);     /* データ書き込み */
  }
  INT DAO Count++;
  if ((INT DAO Count&((INT RestartCount)-1))==0) {
                                   /* バッファサイズをオーバーしたか? */
    INT ReadBufferPtr = INT ReadBufferTop;
  }

  outp(IO Address+0,c);                   /* 割り込みクリア */
  return;
}

/* ディジタルオーディオ入出力割り込み処理 */
void PCI Int waveIn(void)
{
  int i;
  unsigned char c;
  unsigned long l;
  c = inp(IO Address+0);                  /* ステータス確認 */
  /* DAI入力データレディ割り込み */
  for(i=0;i<128;i++) {                    /* 128ワード読み出し */
    l= IoReadLong(IO Address+0x18);       /* データ読み出し */
     IoWriteLong(IO Address+0x18,l);      /* DAOデータ書き込み */
    *INT WriteBufferPtr++ = l;            /* 書き込みバッファ */
  }
  INT DAI Count++;
  if ((INT DAI Count&((INT RestartCount)-1))==0) {
                                   /* バッファサイズをオーバーしたか? */
    INT WriteBufferPtr = INT WriteBufferTop;
  }

  outp(IO Address+0,c);                   /* 割り込みクリア */
  return;
}
                    ～中略～
/* waveファイルのロード準備
   RIFFヘッダを解析して情報を取得する */
int waveLoadPrepare(void)
{
  char buf[256];
  unsigned long n, i;
  /* FILE *fp; */
  int fh;
  fh = global.fh;
  /* R I F F */
  n = read(fh, buf, 4);
  dump(buf, n);
  if(n != 4) return -1;
  if(strncmp("RIFF", buf, 4) != 0) return -2;
  /* get fileSize(-8) */
  n = read(fh, buf, 4);
  dump(buf, n);
  if(n!=4) return -1;
  global.fileSize = *(unsigned long *)(&buf[0]);
  /* W A V E */
  n = read(fh, buf, 4);
  dump(buf, n);
  if(n!=4) return -1;
  if(strncmp("WAVE", buf, 4) != 0) return -2;
  /* read chunk */
  while(1){
    n = read(fh, buf, 4);
  dump(buf, n);
    if(n!=4) return -1;
    if(strncmp("fmt ", buf, 4) == 0) {
      printf("chunkID: fmt ¥n");
      /* found 'fmt ' */
      n = read(fh, buf, 4);
  dump(buf, n);
      if(n!=4) return -1;
      global.fmt.chunkSize
           = ((*(unsigned long *)(&buf[0]))+1) & 0xfffffffeL;
      i = global.fmt.chunkSize;
      if(i>16) i = 16;
      n = read(fh, &global.fmt.wFormatTag, i);
  dump((char *)&global.fmt.wFormatTag, n);
      if(n != i) return -1;
      /* skip */
      i = global.fmt.chunkSize - i;
      if(i>0) {
        lseek(fh, i, SEEK CUR);
        printf("skip %d bytes¥n", i);
      }
    } else
    if(strncmp("data", buf, 4) == 0) {
      printf("chunkID: data¥n");
      /* found data */
      n = read(fh, buf, 4);
  dump(buf, n);
      if(n!=4) return -1;
      global.data.chunkSize = *(unsigned long *)(&buf[0]);
      break; /* 脱出 */

    } else {
      printf("chunkID: %c%c%c%c¥n",
                          buf[0], buf[1], buf[2], buf[3]);
      n = read(fh, buf, 4);
  dump(buf, n);
      if(n!=4) return -1;
      i = ((*(unsigned long *)(&buf[0]))+1) & 0xfffffffeL;
      if(i>0) {
        lseek(fh, i, SEEK CUR);
        printf("skip %d bytes¥n", i);
      }
    }
  }
  if(global.fmt.chunkSize == 0) {
    printf("fatal error: cannot found 'fmt ' chunk.¥n");
    return -1;
  }
  if(global.fmt.wFormatTag != 1) {
    printf("fatal error: cannot play except 'PCM WAVE' format.
                (wFormatTag = %u)¥n", global.fmt.wFormatTag);
    return -1;;
  }
  if(global.forceFrequency != 0){
    global.fmt.dwSamplesPerSec = global.forceFrequency;
  }
  /* 必要なパラメータを取得 */
  n = 0;
  /* サンプリングレートをチェックする */
  while(rates[n] != 0){
    if(global.fmt.dwSamplesPerSec == rates[n]){
      /* 該当するサンプリングレートに対応するrate, factorを設定する */
      global.rate = rates rate[n];
      global.factor = rates factor[n];
      break;
    }
    n++;
  }
  if(global.rate == 0){
    printf("SamplesPerSec not supported.(%lu)¥n",
                              global.fmt.dwSamplesPerSec);
    return -1;
  }

  if((global.fmt.wChannels<1) ||(global.fmt.wChannels>2)){
    printf("Channels not supported.(%u)¥n",
                                    global.fmt.wChannels);
    return -1;
  }
  global.channels = global.fmt.wChannels;
  if((global.fmt.wBitsPerSample != 8) &&
(global.fmt.wBitsPerSample != 16)) {
    printf("BitsPerSample not supported.(%u)¥n",
                               global.fmt.wBitsPerSample);
    return -1;
  }
  global.bits = global.fmt.wBitsPerSample;
  /* check chunkSize */
  if(global.data.chunkSize == 0){
                    ～以下略～
```

音楽配信技術の最新動向

音楽配信技術について
―― Ogg Vorbis

岸　哲夫

映像・音楽の分野でも，インターネットによって情報を発信することがますます重要視されています．ストリーミング配信などの手段によって，無料または課金された音楽や映像が日々配信されています．レンタルビデオを借りるように，オンデマンドでビデオを見ることも可能になりました．

この連載では，それらを実現させるための周辺技術について解説していきます．大きく分けて，以下の事柄を解説していきます．

- 音楽配信技術について
- 音楽を配信する側の現状
- OpenAudioLicenceについて
- 音楽配信技術についての概略
- 映像配信技術について
- 映像を配信する側の現状
- PDA・携帯を使った映像配信について
- ストリーミング配信技術の概略
- オープンソフトウェアで映像・音楽を作り配信する手段

さて，ネット上のCD販売ショップでは，かつて商品の視聴をさせるために，MP3フォーマットの楽曲をダウンロードできるようにしてありました．

しかし現在，ライセンスの問題が発生したため，MP3は少なくとも商用では見かけることがなくなりました．たとえば，有名な「タワーレコード」のWebサイトでは視聴用のフォーマットに「WMA」を使用しています．また，「HMV」ではMAGIQLIPというソフトをインストールして視聴させるシステムになっています．

MP3フォーマットは，ドイツのFraunhofer IIS-A社が作成しました．そして，数年前からネット上に急速に普及したMP3フォーマットに対して，Fraunhofer IIS-A社が使用料を請求し始めたのです．そうなると，MP3のエンコーダを作ったり曲を配信するとライセンス料を支払わなくてはなりません．そこでOgg Vorbisというフォーマットが脚光を浴びています．MP3より優れた圧縮率と音質，そしてオープンソフトウェアであるということで採用されています．

今回は，Ogg Vorbisフォーマットについて少し詳しく解説します．

Ogg Vorbisの公式サイトを**図1**に示します．

2002年の7月にバージョン1.0が正式にリリースされました．

これはOggプロジェクトという名称のツール群で，Vorbisという音声圧縮ツールということだそうです．

いままでにも音声圧縮技術はいくつかありました．たとえば，初期のQuickTimeは，映像を取り扱うよりも音声の圧縮に使われることのほうが多くありました．そして，その後にMP3が一般化しました．

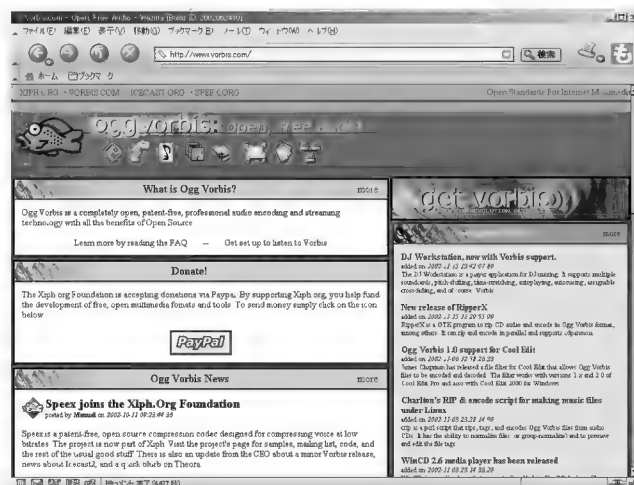
このVorbisは，GNU GPL(GNU General Public License)の元で使用できます．ライブラリとSDKはLGPLライセンスです．もちろん，ソースもWebサイトにあるので，参照することが可能です．

今回は，VorbisのWebサイトにある技術文書を参考にして解説します．

● 原音の再現性について

原音の再現性について，Vorbisの開発者たちは次のように語っています．

> *原音の再現性に関しては機械的な測定基準に基づいて大いに主観的です．ディジタルアンプを通したCDの音声と真空管アンプを通したLPレコードの音声のどちらを選ぶかは，個々の感覚です．測定した歪率で比較したところで，*

〔図1〕Ogg VorbisのWebページ(`http://www.vorbis.com/`)

あまり意味をなしません.

CODECであるASPEC, ATRAC, MP3, WMA, AAC, TwinVQ, AC3, およびVorbisは, ビットレートを減少させるために, 客観的忠実度を失います. これは事実です.

ビットレートを増加させて, 圧縮率を減らすこと, またその逆のどちらを選ぶか悩まなくてはなりません.

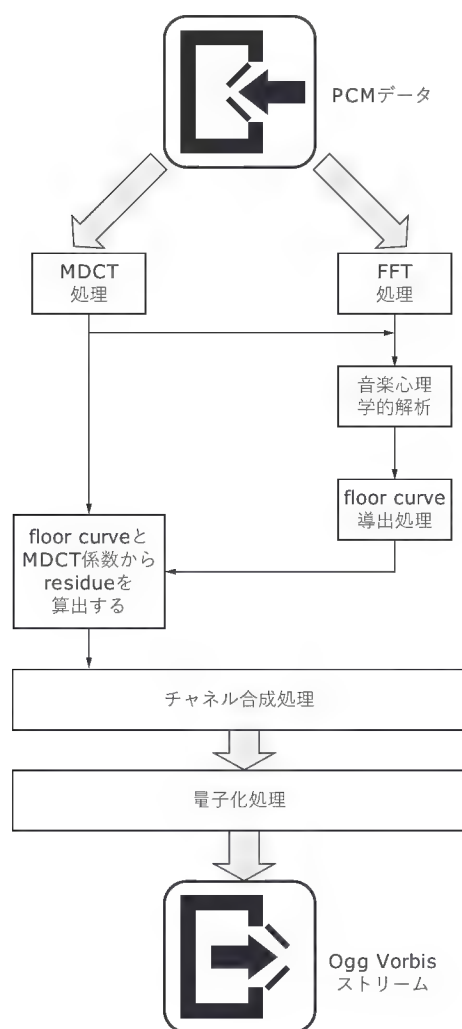
理想は, 忠実度の損失を少なくしもっとも良いトレードオフを選択することです.

FMラジオの大部分の聞き手は, そのメディアがコンパクトディスク, またはDATと比べ, どのくらいのさらに低い忠実度であるかのかなどと感じたりはしません.

しかし, 測定結果と比較されたとき, 差異は, 明白です.

カセットテープの忠実度はもっと低いけれども, 容易に扱えます.

低ビットレートでコード化された平均44.1kHzステレオのMP3とカセットを比較すると, MP3は実際には高い客観的忠実度でしょう. しかし, 主観的にはるかに悪いように聞こえます.

〔図2〕
概略処理フロー

このように, ユーザーの要求を満たすためにCODECが主観的品質を犠牲にしなければならないとき, その結果は, 一般に比較なしで気付きにくい, あるいは無視しやすい差異であるべきです. 明らかに人工的な音質は望まれません.

主観的忠実さを犠牲にすることが必要であるとき, その目標は, 明らかに人工品ではなく差異をめざして努力することです.

CODECの圧倒的大多数は, 必ずあるレベルまでクオリティを低下させます.

主観的品質を犠牲にすることが必要であるとき, そのような問題を回避することが, Vorbisの基本的な設計思想であり, そしてそれはVorbisの設計, およびチューニングに反映されます.

● Vorbisアプリケーションの技術的特色

Vorbisは, ビットレートの非常に広い範囲にわたって柔軟に可変ビットレートだけを採用する多目的なオーディオCODECです.

MP3は可変ビットレート, 固定ビットレートのどちらにも対応しつつありますが, 多くのデコーダ/エンコーダは, どちらかというと非効率的な固定ビットレートにしか対応していないことがあります.

また, Vorbisは48kbpsで高品質のCD, そしてDATデータをリサンプリングなしでエンコードします. そして8kHzの電話通信から192kHzのディジタルマスタまで, モノラルから5.1チャネルまで対応します.

可変ビットレート(VBR)とは, 高音質が必要な部分では高いビットレートを使用し, それほど高い音質が必要ではないところではビットレートを低くする方式です.

なお, 可変ビットレートだけしか使用できない難点は, ストリーミングや動画の音声として使うことができないという点です.

音楽の密度は一定ではありません. たとえば, 楽器の音が1種類だけ流れている部分では, ビットレートを低くしても問題ない場合があります.

ビットレートが低ければ音質が下がってしまいますが, 同時にファイルサイズは小さくなります. MP3と同程度の音質ではファイルサイズがより小さくなります.

一般に, MP3ではサンプリングが128kbpsあれば満足がいくだけの品質が得られるとされていますが, Vorbisは同程度のファイルサイズで, もう1ランク上の160kbpsサンプリング音質が扱えます.

この点に関しては, 筆者も実際に音質を検証してみました.

音声圧縮の品質は音楽の種類に大きく関係します. 合成音だけが使われるテクノ系の音楽では, 圧縮率が高くなるようです. しかし, 民族音楽などの複雑な生楽器を多用した場合, 前面に出ている楽器音はともかく, 微妙に鳴り響く音は明らかに省略されてしまう場合もありますが, Vorbisでは, その他のCODEC

と比較しても，その度合いが少ないように感じました.

● Ogg Vorbisの処理概略

Vorbisは MP3などと同じく非可逆圧縮を採用しています．ここでは，Vorbisがどのような処理を行っているのかを説明します(**図2**).

● MDCT

MDCTとは，変形離散コサイン変換のことです．DCTだけでは変換ブロックの境界ごとに量子化誤差が発生するため，音の品質に悪影響を与えます．MDCTは，前後の隣接フレームと半分ずつ分析フレームを重ねあわせることによって，この歪みを防いでいます.

● FFT

Vorbisは，MDCTとFFTを使用して，その結果を平均化させています.

●音響心理学的解析

音響心理学に基づいて音声データを解析しています．たとえば，可聴音域から外れた部分の音質を犠牲にしたり，大きな音の後の小さな音を犠牲にして，圧縮率を高めます.

聴覚によって感知できる音との最小レベルと周波数の関係は既知のものです．個々の差はあっても最小可聴限界値はほぼ同じです．それ以上は人が聞くことができ，それ以下は人が聞くことができません.

そこで最小可聴限界以下の部分は人には聞こえないということで，その部分のデータを削除し，圧縮効率を高めます.

また，マスキング特性も利用します．マスキング特性とは，ある大きな音が存在している場合，その音の周波数的に近い音，時間的に近い音は，マスキング効果により聴覚で知覚されないため，最小可聴限界と同様に，聞こえない部分を削除することによりデータを削減して圧縮効率を高めます.

その後，ノイズのマスキング，ハフマン符号ベクトル量子化を行い，ドルビー準拠のチャネル合成を行い，圧縮データが完成します.

● 対応エンコーダについて

Linux対応のエンコーダとして，CRIPというツールがあります．このツールにはリッパー機能もありますが法的にグレーなので，それにはふれません．**図3**にCRIPのWebサイトを示します.

そのほかにもエンコーダはありますが，Linux環境で動作するものはすべて，

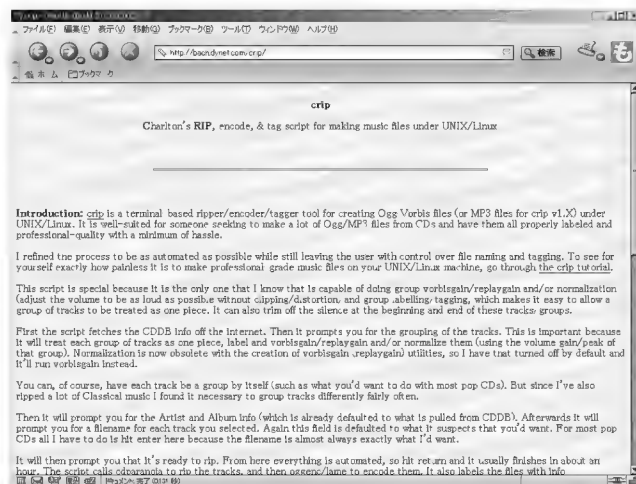
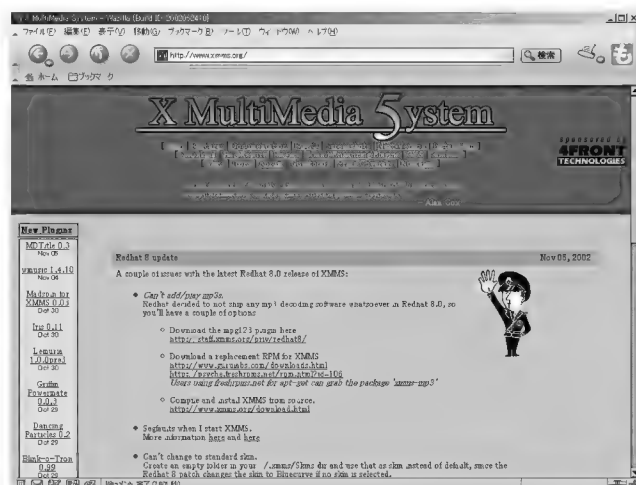
`http://www.vorbis.com/download_unix.psp`

にあるライブラリのrpmパッケージ，またはtarボールをインストールしなければ動作しません.

また，Windowsでは次のツールが使用できます．これはVorbisのWebサイトからダウンロードできます.

Oggdrop - Simple drag and drop GUI encoder;
OggdropXPd - More powerful, more complex drag and drop GUI encoder

〔図3〕CRIPのWebページ(`http://bach.dynet.com/crip/`)

〔図4〕xmmsのWebページ(`http://www.xmms.org/`)

これらはGUIインターフェースをもちます．そして次のツールは，コマンドラインで動作します.

vorbis-tools - command line utilities including
oggenc, ogginfo, vorbiscomment, vcut, and oggdec

● Vorbis対応のソフトウェアプレーヤ

Linux対応のプレーヤとしては，たいていのディストリビューションに入っているXMMSのプラグインで対応しています(**図4**)．また，zinfもプラグインで対応しています(**図5**).

また，Windowsでは，Winamp(**図6**)とSONIQUE(**図7**)が対応しています.

そしてさらに，MacintoshやBeOs，そしてプレイステーション2でも動作するようです．先ごろMacintoshでOgg Vorbis decorder pluginをインストールすることによって，iTunesでOgg Vorbisを使用できるようになったようです.

詳しくは，

`http://www.vorbis.com/download.psp`

を参照してください.

〔図5〕zinfのWebページ（http://www.zinf.org/）

〔図6〕WinampのWebページ（http://www.winamp.com/）

〔図7〕SONIQUEのWebページ（http://sonique.lycos.com/）

〔図8〕tkcPlayerのWebページ（http://www.thekompany.com/embedded/tkcplayer/）

● Vorbis対応のハードウェアプレーヤ

現在，Vorbisに対応している単体のプレーヤはありません．しかし，今後，香港のフロンティア・ラボと韓国のアイリバー社が対応するかもしれないとのことです．

少し意味が違いますが，Linux対応のZaurus上で動作するtkcPlayerがVorbisに対応しています．いち早く野外でVorbisフォーマットの音楽を聞きたいというマニアックな人にはお薦めかもしれません（**図8**）．

● Ogg Vorbisフォーマットの今後

前述のとおり，携帯プレーヤでもVorbisへの対応が考えられているようです．通常，このような製品は「MP3プレーヤ」と呼ばれて，せいぜいWindows規格のWMAフォーマットをサポートしているだけでしたが，今後は「MP3プレーヤ」と呼ばれなくなるかもしれません．

また，現実にはまだ対応していませんが，リアルネットワークスがVorbisと提携すると発表しました．『RealOne Player』と『Helix Universal Server』で対応する予定だそうです．

その企画が始動すると，インターネット上での事実的な標準だったMP3フォーマットも，その地位を大きく揺るがされることになるでしょう．リアルネットワークスが戦略的にオープンソース化を進めていることに大きく関わっていると思います．

そして，音楽を発信する側のミュージシャンたちは，比較してみると音質の良くないMP3ファイルよりVorbisフォーマットを歓迎しているようです．その理由は，Vorbisも同じくロスの多い圧縮形式（非可逆圧縮）ですが，よりダメージを減らすための優れたアルゴリズムを採用しているため，同サイズならMP3ファイルよりも優れた音質で音楽を提供できるからです．

そして，何よりライセンス問題があります．ミュージシャンがネット上でMP3形式で楽曲を販売した場合，Fraunhoferに対し，セールスに応じた一定のロイヤリティを支払う責任を負うことになるでしょう．しかしVorbisの場合，特許もライセンスもないので，販売する場合も，他人に配る場合も，ストリー

ム配信する場合も，面倒な手続きは必要ありません．そういう点でもミュージシャンたちに注目されています．

最新情報として2002年12月6/7日に，大阪で「関西オープンソース＋フリーウェア2002」というイベントがありました．執筆時点で詳細は不明ですが，次回以降でふれる予定の「オープンオーディオライセンス」について，DADAや4-D，P-MODELに参加していた小西健司氏が，デモを交えて講演を行ったようです（**図9**）．

このように，Ogg VorbisはMP3の「代替品」というものではなく，より優れたシステムといえます．

次回は，引き続きOgg Vorbis公式サイトにある技術資料の解説を行います．そして次回以降で，配信する側がどのようにOgg Vorbisを考えているか，そしてOpenAudioLicenseとの関係を検証したいと考えています．

きし・てつお

〔図9〕関西オープンソース＋フリーウェア2002
（http://of.good-day.net/event.html）

やり直しのための信号数学 第14回

FFTによる信号処理応用(システム設計編 I)

三谷政昭

これまで3回にわたり，数学関数編，データ処理編と題して，FFTによる信号処理応用について具体例を示し，FFTの能力の一端を実感してもらった．

今回からは，FFTを用いてディジタルシステムを設計する事例について解説する．まずは，ディジタルシステムの周波数特性，すなわち伝達関数の近似設計におけるFFTの適用の仕方，考え方を中心にわかりやすく説明する．具体的なディジタルシステムとして，ディジタルフィルタを採り上げ，周波数特性のさまざま形状(ローパス，ハイパス，バンドパス，バンドストップ)の例を示す．(筆者)

ディジタルシステムの周波数応答(振幅，位相)

まず，ディジタルシステムの周波数応答(あるいは周波数スペクトル)について簡単にまとめておこう．ディジタルシステムの中で，ディジタルフィルタとよばれるものは雑音を含む信号から雑音を取り除いて信号成分のみを取り出したり，信号の形状を変えたり，信号を平均化したり，信号をいくつかの周波数成分に分解したり，……と，いろいろな信号処理機能をもつものである(**図14.1**)．

ところで周波数応答には，たとえば"振幅"，"位相"とよばれるパラメータがあり，それぞれ次のような性質を表す．

● 振幅特性(**図14.2**)

いま，入力に大きさ2[V]の正弦波信号を加えたときに出力で1.6[V]の正弦波信号が得られたとしよう．このとき，ディジタルシステムの振幅は「0.8(= 1.6/2)である」という．同様に，入力信号の大きさが2[V]で出力信号の大きさが2[V]のときはディジタルシステムの振幅は「1である」といい，入ってきた信号が大きさの変化がなく，そのままの大きさで出力される．このような周波数は"通過域(パスバンド：passband)"とよばれる．

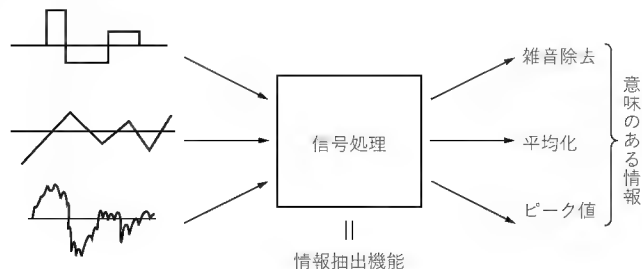
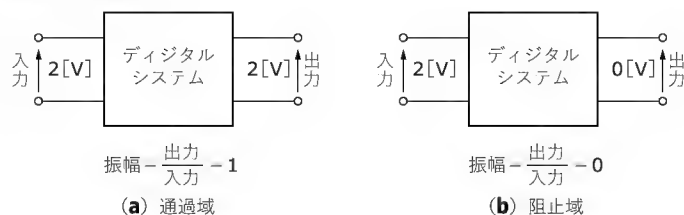
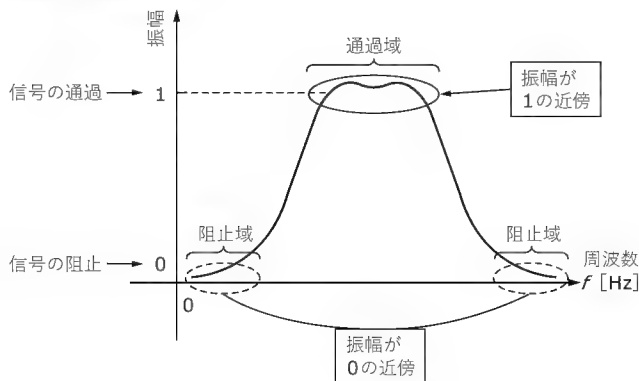
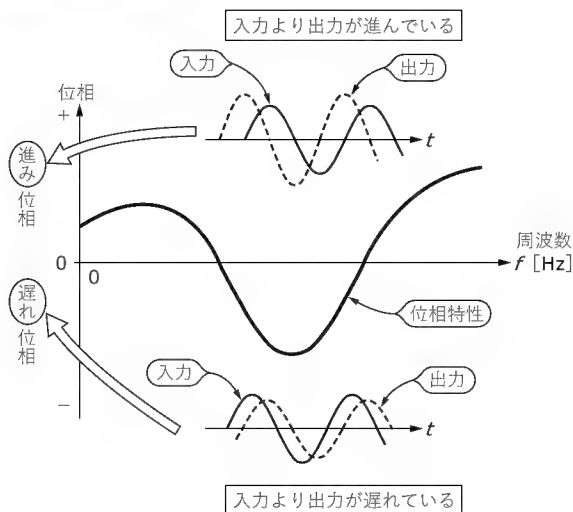
これに対して，入力信号の大きさが2[V]で出力信号の大きさが0[V]のときはディジタルシステムの振幅は「0である」といい，入ってきた信号がまったく出力されないわけで，"阻止域(ストップバンド：stopband，あるいは減衰域)"とよばれる．このように入力信号の大きさに対して，

「どれくらいの割合で信号が出力されるのか」

を表すものが振幅であり，正弦波信号の周波数に対する変化のようすをグラフにしたものが"振幅特性"，あるいは"振幅スペクトル"とよばれている(**図14.3**)．

● 位相特性

さて，入力信号をディジタルシステムに加えると間髪をおかず瞬時に出力が得られると思われるかもしれないが，そうではない．たとえば，国際電話や衛星放送などで外国に住む人との会話を聞いていると，なんとなく間延びした感じで受け答えが

〔**図14.1**〕信号処理とは

〔**図14.2**〕振幅とは

〔図14.3〕振幅特性

〔図14.5〕位相特性

スムーズでないと思うことがあるが，この現象は言葉を発してから届くまでに時間がかかって遅れているためである．ディジタルシステムでも同様で，通常は**図14.4**に示すようにある時間だけ遅れて出力信号が得られることになる．この時間遅れに関係する量が“位相”とよばれるものである．入力信号がディジタルシステムに入ってからどれだけの時間ずれがあって信号が出力されるのかに関係する位相が，正弦波の周波数に対してどのように変化をするのかをグラフにして表したものが，“位相特性”，あるいは“位相スペクトル”とよばれている（**図14.5**）．

たとえば，10[Hz]の周波数の正弦波が0.01[秒]遅れて出力されたときには，

$$-2\pi \times 10 \times 0.01 = -0.2\pi \text{ [rad]}$$

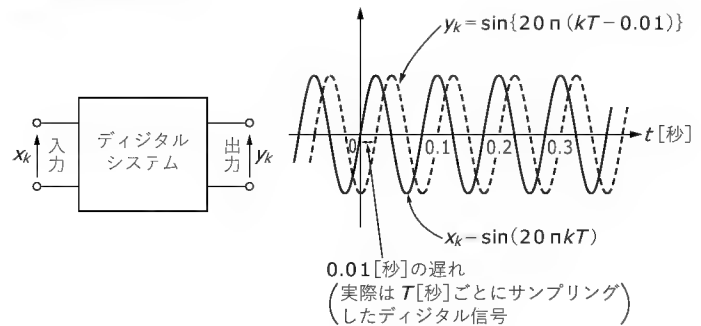
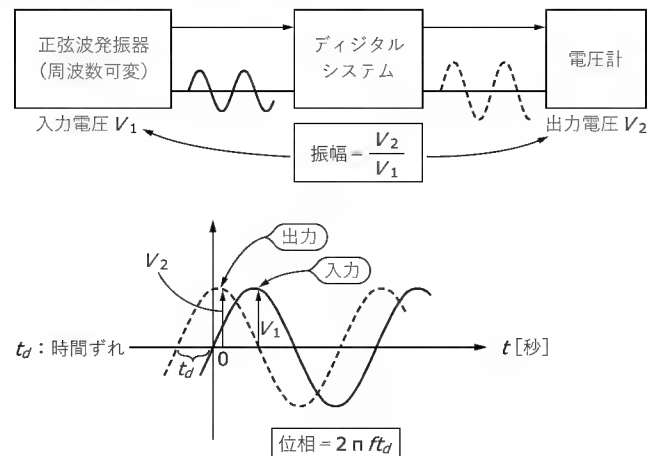
という位相を有するといい，負（マイナス）の符号は時間の“遅れ”を表す．一般的には，入力信号の周波数をf[Hz]，時間の遅れをt_d[秒]とすると，その位相は，

$$-2\pi f t_d \text{ [rad]} \quad \cdots\cdots (1)$$

と表されることになる．

● 周波数応答の測定方法

もっともシンプルな測定方法を**図14.6**に示す．

〔図14.4〕位相とは

〔図14.6〕周波数特性の測定方法

1）周波数を可変できる正弦波発振器を用意する
2）ある周波数f[Hz]の正弦波をディジタルシステムに入力信号として加え，出力信号の大きさを測定する．同時に，オシロスコープ上に入出力波形を表示させ，時間の差t_dを測定する
3）横軸に発振器の周波数fを，縦軸に振幅，あるいは位相をとり，発振器の周波数fを変えながら，それぞれの値をプロットする．その際，振幅と位相は次式で計算して求める．

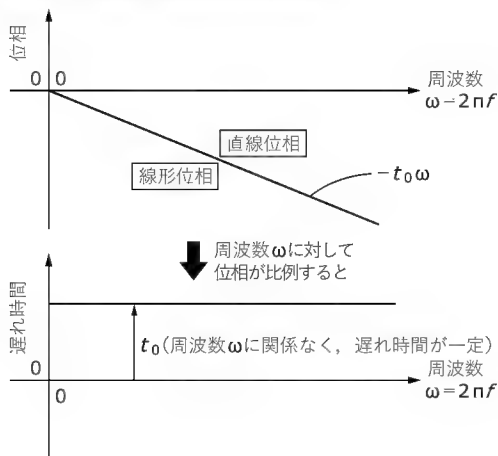
$$\text{振幅} = \frac{\text{出力電圧の大きさ}}{\text{入力電圧の大きさ}} \quad \cdots\cdots (2)$$

$$\text{位相} = 2\pi \times (\text{時間のずれ}) \quad \cdots\cdots (3)$$

なお，位相を角周波数で割った値は“遅延時間”とよばれ，周波数に対して一定値となる特性を“線形位相（あるいは，直線位相）”という（**図14.7**）．

ディジタルシステムの周波数特性例

周波数を変えて得られるディジタルシステムの周波数特性には，通過域（信号を通しやすい周波数領域）と阻止域（信号を通しにくい周波数領域）の組み合わせによって，4種類のディジタルフィルタに大別される（**図14.8**）．ここで，フィルタ（filter）という言葉は，文字どおり信号を選択するという意味であり，ド

〔図 14.7〕線形位相(直線位相)特性

リップコーヒーを入れるときに用いる「ろ紙」の役目と同じような機能を有している．

1) ローパスフィルタ〔**図 14.8(a)**〕

低域通過型フィルタともいう．通過域が角周波数 $\omega=0$[rad](直流に相当)から ω_0 までで，ω_0 から $\omega_s/2$ の範囲が阻止域となり，周波数が高くなるにしたがって通りにくくなり，出力信号が小さくなる．ここで，ω_s はサンプリング周波数で，サンプリング間隔 T[秒]の逆数($=1/T$)に等しい．つまり，ディジタルシステムのもっとも高い周波数は，サンプリング角周波数 ω_s の 1/2 であるということに注意してもらいたい．

2) ハイパスフィルタ〔**図 14.8(b)**〕

高域通過型フィルタともいい，ローパスフィルタの逆の特性を有する．すなわち，角周波数 $\omega=0$[rad](直流に相当)から ω_0 までが阻止域で，ω_0 から $\omega_s/2$ の範囲が通過域となり，周波数が高くなるにしたがって通りやすくなり，出力信号が大きくなる．

3) バンドパスフィルタ〔**図 14.8(c)**〕

帯域通過型フィルタともいい，角周波数 ω_1 から ω_2 までの範囲が通過域で，それ以外の角周波数の範囲は阻止域となる．つまり，中間の周波数はそのまま出力され，低い周波数と高い周波数の信号は減衰してほとんど出力されない．

4) バンドストップフィルタ〔**図 14.8(d)**〕

帯域阻止型フィルタ，あるいはバンドエリミネーションフィルタともいう．角周波数 ω_1 から ω_2 までの範囲が阻止域で，それ以外の角周波数の範囲は通過域となる．つまり，低い周波数と高い周波数の信号はそのまま出力され，中間の周波数は減衰してほとんど出力されない．

図 14.8 に示すように，通過域から阻止域，阻止域から通過域へと非常に急峻な階段状の変化をするような振幅特性を有するディジタルフィルタを実現することは不可能であり，**図 14.9** に示すような実現可能な特性を設計対象のモデルとして考えなければならない．

ディジタルシステムの周波数特性と DFT 値

それでは，ディジタルフーリエ変換(DFT)値とディジタルシステムの伝達関数表現で用いる“z 変換”との関係についての説明を始める．

いま，周期性を有するアナログ信号(**図 14.10**)をサンプリング間隔 T[秒]ごとにサンプリングして得られた N 個のディジタル信号のサンプル値を $\left\{f_n\right\}_{n=0}^{n=N-1}$ とするとき，DFT 値 $\left\{F_k\right\}_{k=0}^{k=N-1}$ は次式で与えられる．

$$F_k=\frac{1}{N}\sum_{n=0}^{N-1}f_n\left(W_N\right)^{nk};k=0,\ 1,\ 2,\ \cdots,\ N-1 \quad \cdots\cdots\cdots(4)$$

〔図 14.8〕理想的なフィルタの振幅仕様

(a) ローパスフィルタ

(b) ハイパスフィルタ

(c) バンドパスフィルタ

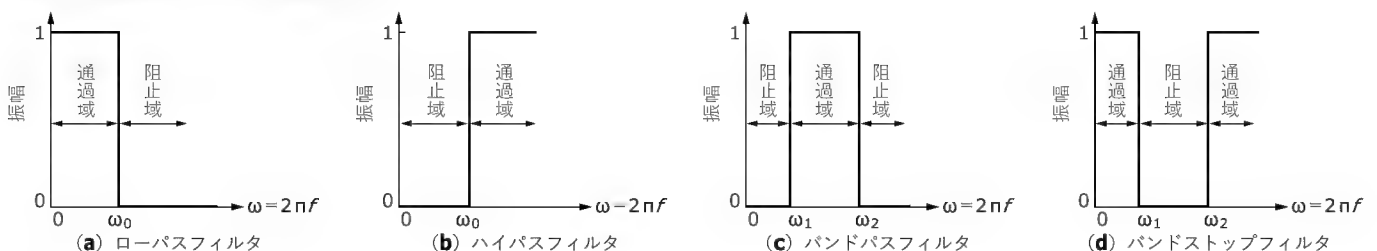
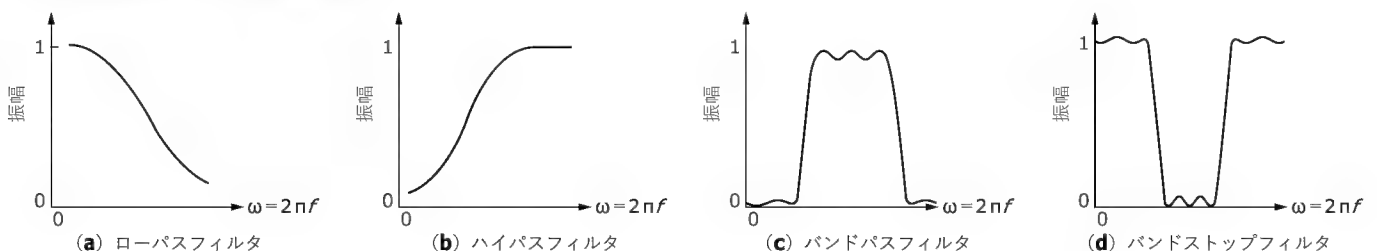
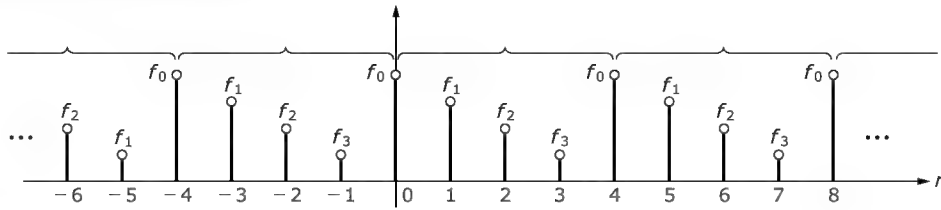
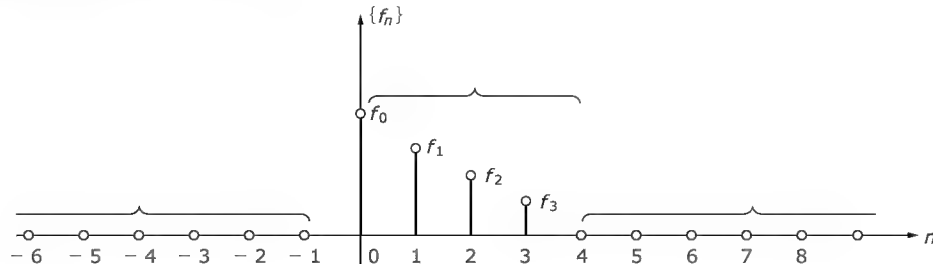
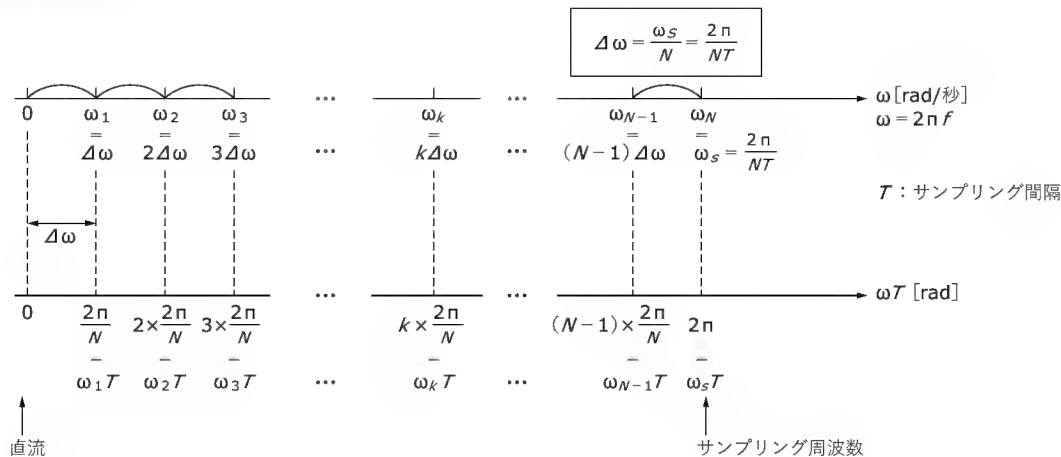
(d) バンドストップフィルタ

〔図 14.9〕実際のフィルタの振幅特性例

(a) ローパスフィルタ

(b) ハイパスフィルタ

(c) バンドパスフィルタ

(d) バンドストップフィルタ

〔図14.10〕周期性を有するディジタル信号の例（$N=4$）

〔図14.11〕時間制限されたディジタル信号の例（$N=4$）

〔図14.12〕周波数分解能$\Delta\omega$

ただし，$W_N = e^{-j2\pi/N}$ ……………………(5)

また，**図14.11**のディジタル信号系列$\{f_n\}_{n=0}^{n=N-1}$のz変換は，

$$G(z)=\sum_{n=0}^{N-1} f_n z^{-n} \quad \cdots\cdots(6)$$

であり，$z=e^{j\omega T}$を代入して周波数（スペクトル）特性が計算される．

$$G\left(e^{j\omega T}\right)=\sum_{n=0}^{N-1} f_n e^{-jn\omega T} \quad \cdots\cdots(7)$$

角周波数ωは0〜ω_s（$\omega_s=2\pi/T$[rad/秒]，サンプリング角周波数）の範囲の値をとり，N等分した各周波数をω_kとすると，

$$\omega_k = k\Delta\omega \ ; \ k=0,\ 1,\ 2,\ \dots,\ N-1 \quad \cdots\cdots(8)$$

ただし，$\Delta\omega=\dfrac{\omega_s}{N}=\dfrac{2\pi}{NT}$

となり，$\Delta\omega$は周波数分解能に相当する（**図14.12**）．

ここで，$\omega=\omega_k$における$G(z)$の値$G\left(e^{j\omega_k T}\right)$を$G_k$とし，式(8)より得られる$\omega_k T=k(2\pi/N)$を式(7)に代入すると，

$$G_k = G\left(e^{j\omega_k T}\right) = \sum_{n=0}^{N-1} f_n e^{-jnk(2\pi/N)} = \sum_{n=0}^{N-1} f_n \left(e^{-j2\pi/N}\right)^{nk}$$

となる．さらに，式(5)のW_Nを用いて書き換えると，

$$G_k=\sum_{n=0}^{N-1} f_n W_N{}^{nk} \quad \cdots\cdots(9)$$

なる関係が導かれる．

よって，式(4)と式(9)を比較することにより，以下の関係式が得られる．

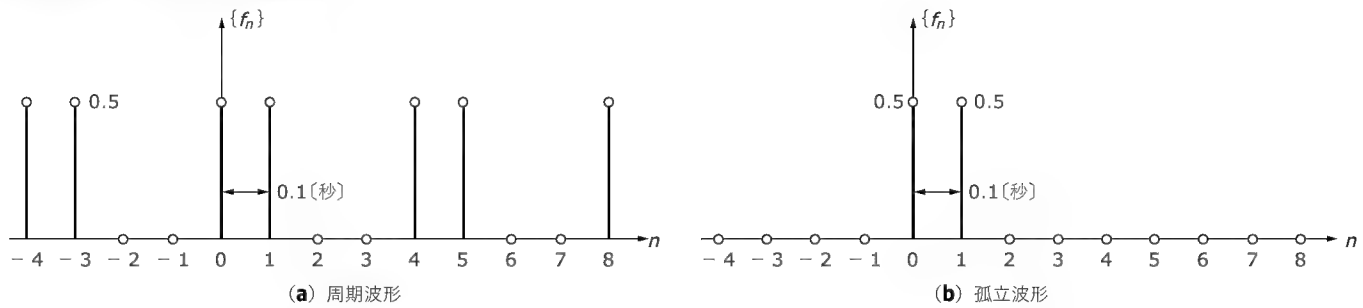
$$F_k=\frac{1}{N}\times G_k \quad \cdots\cdots(10)$$

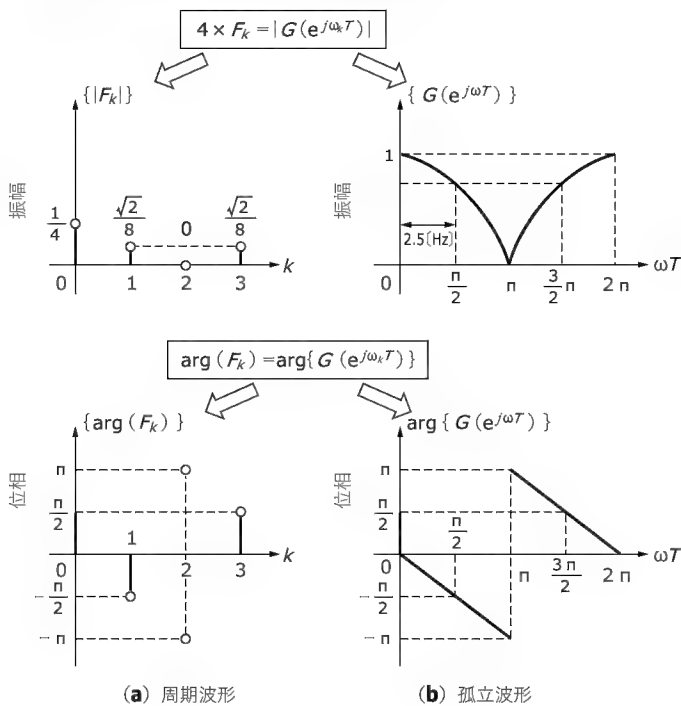
$$G_k=N\times F_k \quad \cdots\cdots(11)$$

つまり，z変換に基づく周波数スペクトル値$\{G_k\}_{k=0}^{k=N-1}$がDFT

〔図 14.13〕ディジタル信号 例題1

(a) 周期波形

(b) 孤立波形

〔図 14.14〕周波数応答の振幅，位相 例題1

(a) 周期波形 (b) 孤立波形

値 $\{F_k\}_{k=0}^{k=N-1}$ を N 倍したものに相当するわけで，式(4)の$(1/N)$倍を除いて計算した結果に一致することがわかる．

例題1

図 14.13(**a**)，(**b**)に示すディジタル信号の周波数成分を求めよ．

解答1

図 14.13(**a**)の周期波形は，$N = 4$ として式(4)を計算すると，周波数成分は離散的な形で次のように求められる〔**図 14.14**(**a**)〕．

$$F_0 = \frac{1}{4}(f_0 + f_1 + f_2 + f_3) = \frac{1}{4}(0.5 + 0.5) = \frac{1}{4}$$

$$F_1 = \frac{1}{4}(f_0 - jf_1 - f_2 + jf_3)$$

$$= \frac{1}{4}(0.5 - j0.5) = 0.125 - j0.125 = \frac{1}{8}(1 - j)$$

$$F_2 = \frac{1}{4}(f_0 - f_1 + f_2 - f_3) = \frac{1}{4}(0.5 - 0.5) = 0$$

$F_3 = F_1$ の複素共役

$$= 0.125 + j0.125 = \frac{1}{8}(1 + j)$$

ここで，サンプリング間隔 T が 0.1［秒］なので，サンプリング周波数が，

$$f_s = \frac{1}{0.1[秒]} = 10[\mathrm{Hz}]$$

となり，周波数分解能は，

$$\Delta f = \frac{f_s}{N} = \frac{10}{4} = 2.5[\mathrm{Hz}]$$

で与えられる．よって，各周波数(0，2.5，5，7.5［Hz］)に対する振幅 $|F_k|$ と位相 $\arg(F_k)$ は以下のように計算される．なお，数学的には，振幅は絶対値に，位相は偏角に相当する．

(i) $f = 0$［Hz］(直流)

$$|F_0| = \frac{1}{4},\ \arg(F_0) = 0$$

(ii) $f = 2.5$［Hz］

$$|F_1| = \frac{1}{8}\sqrt{1^2 + (-1)^2} = \frac{\sqrt{2}}{8},$$

$$\arg(F_1) = \arctan(1,\ -1) = \tan^{-1}\left(\frac{-1}{1}\right) = -\frac{\pi}{4}$$

(iii) $f = 5$［Hz］

$$|F_2| = 0,\ \arg(F_2) = \pm\frac{\pi}{2}$$

(iv) $f = 7.5$［Hz］

$$|F_3| = \frac{1}{8}\sqrt{1^2 + 1^2} = \frac{\sqrt{2}}{8} = |F_1|,$$

$$\arg(F_3) = \arctan(1,\ 1) = \tan^{-1}\left(\frac{1}{1}\right) = \frac{\pi}{4} = -\arg(F_1)$$

また，**図 14.13**(**b**)の孤立波形の周波数成分は，式(7)を計算して連続的な形で次のように求められる〔**図 14.14**(**b**)〕．

$$G(e^{j\omega T}) = 0.5 + 0.5\,e^{-j\omega T}\ ;\ \omega = 2\pi f$$

$$= 0.5 + 0.5 \times \{\cos(\omega T) - j\sin(\omega T)\}$$

$$= \{0.5 + 0.5\cos(\omega T)\} - j0.5\sin(\omega T) \quad \cdots\cdots(12)$$

このように，周期波形と孤立波形とで周波数成分のようすが

違っていることに注意してもらいたい．つまり，周期波形は離散的，孤立波形は連続的な周波数成分をもつのである．ここで，式(12)において，DFT 値が表す離散的な周波数(0，2.5，5，7.5[Hz])に対する G_k ($k=0$，1，2，3)を求めてみると，

$$G_0 = G\left(e^{j0}\right) = 1$$

$$G_1 = G\left(e^{j2\pi\times 2.5\times 0.1}\right) = G\left(e^{j0.5\pi}\right)$$
$$= \left\{0.5 + 0.5\cos(0.5\pi)\right\} - j0.5\sin(0.5\pi)$$
$$= \frac{1}{2}(1-j)$$

$$G_2 = G\left(e^{j2\pi\times 5\times 0.1}\right) = G\left(e^{j\pi}\right)$$
$$= \left\{0.5 + 0.5\cos(\pi)\right\} - j0.5\sin(\pi) = 0$$

$$G_3 = G\left(e^{j2\pi\times 7.5\times 0.1}\right) = G\left(e^{j1.5\pi}\right)$$
$$= \left\{0.5 + 0.5\cos(1.5\pi)\right\} - j0.5\sin(1.5\pi)$$
$$= \frac{1}{2}(1+j)$$

となる．よって，F_k ($k=0$，1，2，3)と比べてみると，

$$G_k = 4 \times F_k$$

という，式(10)と式(11)の関係が成立することが確認できる．

FFT によるディジタルシステム設計

それでは，サンプル数 $N=4$[個]に対するディジタルシステムの振幅特性 $\left|G\left(e^{j\omega T}\right)\right|$ が周期性を有することを利用して，FFT を利用して設計する手順を示しておこう(**図 14.15**)．

いま，振幅特性 $\left|G\left(e^{j\omega T}\right)\right|$ を有限項(項数が4個)のフーリエ級数で表すことを考える．すなわち，展開係数を $\left\{h_n\right\}_{n=-1}^{n=2}$ とすれば，

$$G\left(e^{j\omega T}\right) = h_0 + h_1 e^{j\omega T} + h_2 e^{j2\omega T} + h_{-1}e^{-j\omega T} \quad\cdots\cdots(13)$$

と表され，周波数分解能に対応する周波数は，

$$\omega_k T = k\left(\frac{2\pi}{4}\right) = \frac{k\pi}{2}\ ;\ k = 0,1,2,3 \quad\cdots\cdots(14)$$

となる．そこで，まず式(5)より，

$$W = W_4 = e^{-j\frac{\pi}{2}} \quad\cdots\cdots(15)$$

とおき，式(14)を式(13)に代入すると，以下の関係式が得られる．

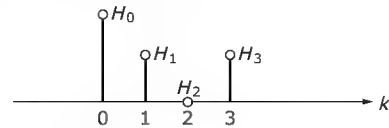
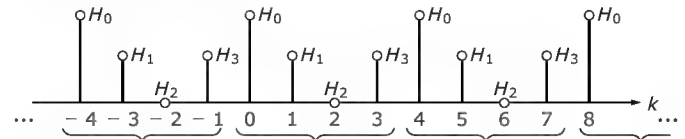
$$H_0 = G\left(e^{j0}\right) = h_0 + h_1 + h_2 + h_{-1} \quad\cdots\cdots(16)$$

$$H_1 = G\left(e^{j\frac{\pi}{2}}\right) = h_0 + h_1 W^{-1} + h_2 W^{-2} + h_{-1} W^{1} \quad\cdots\cdots(17)$$

$$H_2 = G\left(e^{j\pi}\right) = h_0 + h_1 W^{-2} + h_2 W^{-4} + h_{-1} W^{2} \quad\cdots\cdots(18)$$

$$H_3 = G\left(e^{j\frac{3\pi}{2}}\right) = h_0 + h_1 W^{-3} + h_2 W^{-6} + h_{-1} W^{3} \quad\cdots\cdots(19)$$

〔図 14.15〕FFT によるディジタルシステム設計の流れ($N=4$)

① ディジタルシステムの振幅仕様を与える

② 振幅仕様を周期波形とみなす

③ 周期波形のフーリエ級数の展開係数を求める

$\{h_0\ \ h_1\ \ h_2\ \ h_3\}$

h_{-1}

$h_{-1}e^{-j\omega T} + h_0 + h_1 e^{j\omega T} + h_2 e^{j2\omega T}$ (フーリエ級数展開)

④ ディジタルシステムの伝達関数 $H(z)$ に直す

また，式(15)の回転因子 W は，

$$W^4 = 1 \quad\cdots\cdots(20)$$

なので，$W^1 = W^{-3}$，$W^2 = W^{-6}$，$W^3 = W^{-9}$ であり，式(16)〜式(19)に代入して以下のように表される．

$$H_0 = h_0 + h_1 + h_2 + h_{-1} \quad\cdots\cdots(21)$$

$$H_1 = h_0 + h_1 W^{-1} + h_2 W^{-2} + h_{-1} W^{-3} \quad\cdots\cdots(22)$$

$$H_2 = h_0 + h_1 W^{-2} + h_2 W^{-4} + h_{-1} W^{-6} \quad\cdots\cdots(23)$$

$$H_3 = h_0 + h_1 W^{-3} + h_2 W^{-6} + h_{-1} W^{-9} \quad\cdots\cdots(24)$$

ところで，h_{-1} を h_3 とみなせば，式(21)〜式(24)はフーリエ逆変換(IDFT)の計算式に一致していることから，展開係数 $\left\{h_n\right\}_{n=0}^{n=3}$ は $\left\{H_k\right\}_{k=0}^{k=3}$ を入力とする DFT 値によって計算されることがわかる．

一般に，決定すべきフーリエ級数の展開係数が N[個]であるときは，周波数特性を N 等分し，その周波数標本点の DFT 値を求めることによって展開係数が得られる(**図 14.16**)．

以上の計算から得られた展開係数より，ディジタルシステムの伝達関数 $G(z)$ は，式(13)において $z = e^{j\omega T}$ と置き換え，$h_2 \neq 0$ とすれば，

$$G(z) = \left(h_0 + h_1 z^1 + h_2 z^2 + h_{-1} z^{-1}\right) \times z^{-2}$$
$$= h_2 + h_1 z^{-1} + h_0 z^{-2} + h_{-1} z^{-3} \quad\cdots\cdots(25)$$

で与えられ，周波数特性は，$z = e^{j\omega T}$ を代入して，

$$G\left(e^{j\omega T}\right) = h_2 + h_1 e^{-j\omega T} + h_0 e^{-j2\omega T} + h_{-1} e^{-j3\omega T}$$

で求められる(**図 14.17**)．また，$h_2 = 0$ の場合は，

$$G(z) = \left(h_0 + h_1 z^1 + h_{-1} z^{-1}\right) \times z^{-1}$$
$$= h_2 + h_0 z^{-1} + h_{-1} z^{-2} \quad\cdots\cdots(26)$$

と表される．なお，式(26)の変形に際し，それぞれ z^{-2}，z^{-1} をかけているが，ディジタルフィルタの因果性を担保するためのものである．

例題2

図 14.18 のローパスフィルタを設計し，周波数特性(振幅，位相)を示せ．

解答2

図 14.18 より，$H_0 = 1$，$H_1 = 0.5$，$H_2 = 0$，$H_3 = 0.5$ であり，$N = 4$ として式(4)に代入すれば，

$$h_0 = \frac{1}{4}\left(H_0 + H_1 + H_2 + H_3\right) \cdots\cdots (27)$$

$$h_1 = \frac{1}{4}\left(H_0 + H_1 W^1 + H_2 W^2 + H_3 W^3\right) \cdots\cdots (28)$$

$$h_2 = \frac{1}{4}\left(H_0 + H_1 W^2 + H_2 W^4 + H_3 W^6\right) \cdots\cdots (29)$$

$$h_{-1} = h_3 = \frac{1}{4}\left(H_0 + H_1 W^3 + H_2 W^6 + H_3 W^9\right) \cdots\cdots (30)$$

となり，さらに，

$$W = W_4 = e^{-j\frac{\pi}{2}} = -j$$

を考慮して計算することにより，

$$h_0 = 0.5,\ h_1 = 0.25,\ h_2 = 0,\ h_{-1} = 0.25 \cdots\cdots (31)$$

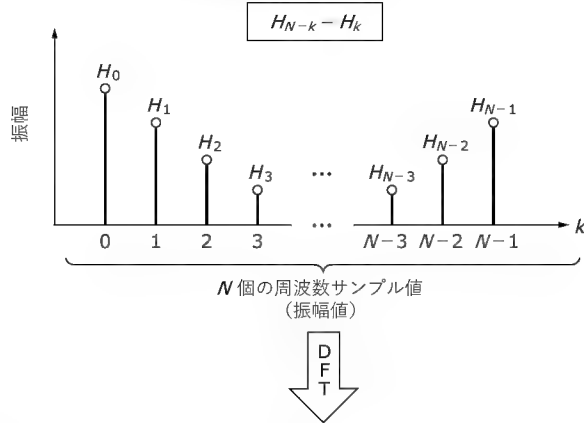
と計算される．よって，式(26)より**図 14.18** のローパスフィルタの伝達関数を $H_{LP}(z)$ と表せば，

$$H_{LP}(z) = 0.25 + 0.5z^{-1} + 0.25z^{-2} \cdots\cdots (32)$$

で与えられ，周波数特性は，$z = e^{j\omega T}$ を代入して，

$$H_{LP}\left(e^{j\omega T}\right) = G\left(e^{j\omega T}\right) = \left\{\cos\left(\frac{\omega T}{2}\right)\right\}^2 e^{-j\omega T} \cdots\cdots (33)$$

で求められる(**図 14.19**)．

〔図 14.16〕N 個のフーリエ級数展開係数の算出

フーリエ級数展開	$h_0 + h_1 e^{j\omega T} + h_2 e^{j2\omega T} + \cdots + h_{N-2} e^{-j2\omega T} + h_{N-1} e^{-j\omega T}$

〔図 14.17〕フーリエ級数展開式から伝達関数の算出($N = 4$)

● フーリエ級数展開

$G(e^{j\omega T}) = h_0 + h_1 e^{j\omega T} + h_2 e^{-j2\omega T} + h_{-1} e^{-j\omega T}$

$e^{j\omega T} = z$ とおく

● 仮の伝達関数

$G(z) = h_0 + h_1 z + h_2 z^2 + h_{-1} z^{-1}$

因果律を満たすように遅延をかける

① $h_2 \neq 0$ の場合は z^{-2} を掛ける

② $h_2 = 0$，$h_1 \neq 0$ の場合は z^{-1} を掛ける

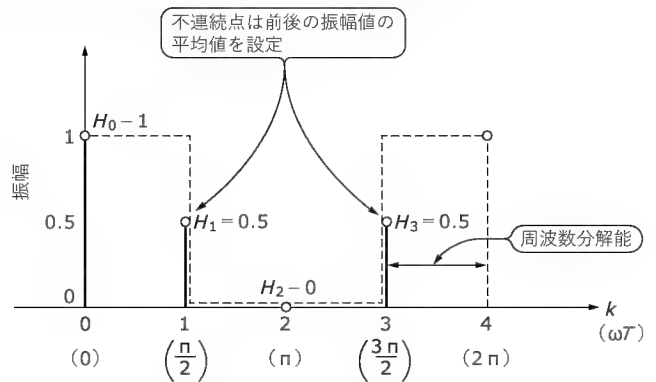
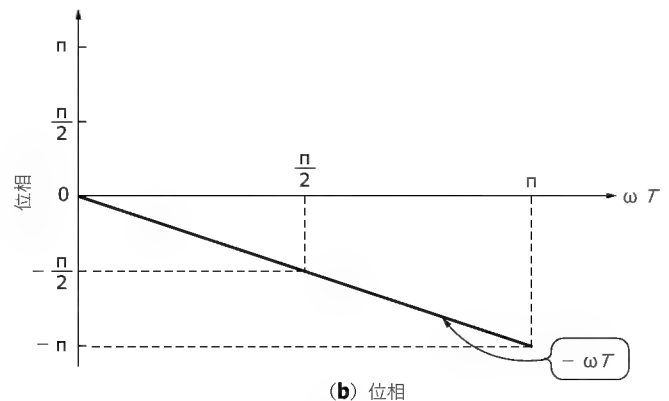
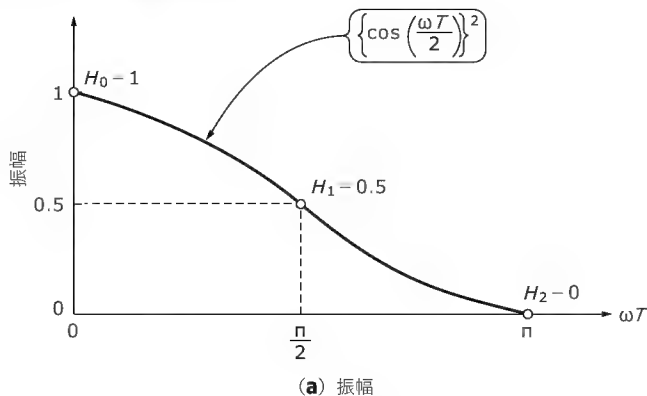
① の場合：$G(z) = h_2 + h_1 z^{-1} + h_0 z^{-2} + h_{-1} z^{-3}$

② の場合：$G(z) = h_1 + h_0 z^{-1} + h_{-1} z^{-2}$

〔図 14.18〕例題2 ローパスフィルタの設計仕様

〔図 14.19〕例題2 の周波数特性

(a) 振幅

(b) 位相

〔図14.20〕 例題3 ハイパスフィルタの設計仕様

〔図14.22〕 例題4 バンドパスフィルタの設計仕様

例題3

図14.20のハイパスフィルタを設計し，周波数特性（振幅，位相）を示せ．また，ローパスフィルタの係数〔式(32)〕と比較せよ．

解答3

図14.20より，$H_0 = 0$，$H_1 = 0.5$，$H_2 = 1$，$H_3 = 0.5$であり，式(4)のDFT値は，

$$h_0 = 0.5,\ h_1 = -0.25,\ h_2 = 0,\ h_{-1} = -0.25 \cdots\cdots (34)$$

と計算される．よって，式(26)より**図14.20**のハイパスフィルタの伝達関数を$H_{HP}(z)$と表せば，

$$H_{HP}(z) = -0.25 + 0.5z^{-1} - 0.25z^{-2} \cdots\cdots (35)$$

で与えられ，周波数特性は，$z = e^{j\omega T}$を代入して，

$$H_{HP}\left(e^{j\omega T}\right) = G\left(e^{j\omega T}\right) = \left\{\sin\left(\frac{\omega T}{2}\right)\right\}^2 e^{-j\omega T} \cdots\cdots (36)$$

で求められる（**図14.21**）．

例題4

図14.22のバンドパスフィルタを設計し，周波数特性（振幅，位相）を示せ．

解答4

図14.22より，$H_0 = 0$，$H_1 = 0.5$，$H_2 = 1$，$H_3 = 0.5$，$H_4 = 0$，$H_5 = 0.5$，$H_6 = 1$，$H_7 = 0.5$であり，$N = 8$として，式(5)より，

$$W_8 = e^{-j\frac{\pi}{4}} = \cos\left(\frac{\pi}{4}\right) - j\sin\left(\frac{\pi}{4}\right) = \frac{1-j}{\sqrt{2}}$$

となり，式(4)に代入してDFT値を求めれば，

$$h_0 = 0.5,\ h_1 = 0,\ h_2 = -0.25,\ h_3 = 0$$
$$h_4 = 0,\ h_{-3} = 0,\ h_{-2} = -0.25,\ h_{-1} = 0 \cdots\cdots (37)$$

と計算される．ここで，$N = 8$のときの伝達関数$G(z)$は，$N = 4$の場合と同様の計算により，

$$G(z) = (h_0 + h_1 z + h_2 z^2 + h_3 z^3 + h_4 z^4 + h_{-3} z^{-3} + h_{-2} z^{-2} + h_{-1} z^{-1}) \times z^{-4}$$
$$= h_4 + h_3 z^{-1} + h_2 z^{-2} + h_1 z^{-3} + h_0 z^{-4} + h_{-1} z^{-5} + h_{-2} z^{-6} + h_{-3} z^{-7} \cdots (38)$$

で与えられ，周波数特性は，

$$G\left(e^{j\omega T}\right) = (h_4 + h_3 e^{-j\omega T} + h_2 e^{-j2\omega T} + h_1 e^{-j3\omega T} + h_0 e^{-j4\omega T} + h_{-1} e^{-j5\omega T} + h_{-2} e^{-j6\omega T} + h_{-3} e^{-j7\omega T}) \cdots\cdots (39)$$

で求められる．

よって，式(37)を式(38)に代入し，**図14.22**のバンドパスフィルタの伝達関数を$H_{BP}(z)$と表せば，$h_3 = h_4 = h_{-3} = 0$を考慮して，

$$H_{BP}(z) = (h_0 + h_1 z + h_2 z^2 + h_{-2} z^{-2} + h_{-1} z^{-1}) \times z^{-2}$$
$$= h_2 + h_1 z^{-1} + h_0 z^{-2} + h_{-1} z^{-3} + h_{-2} z^{-4} \cdots (40)$$

で与えられ，周波数特性は，$z = e^{j\omega T}$を代入して，

〔図14.21〕 例題3 の周波数特性

(a) 振幅

(b) 位相

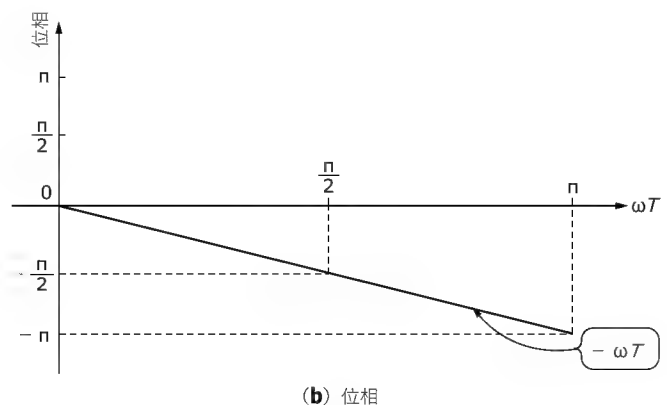
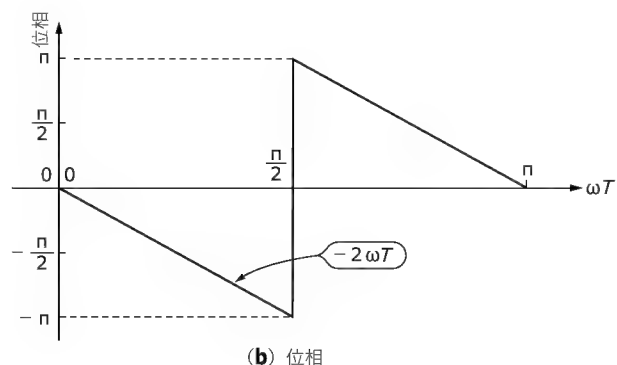
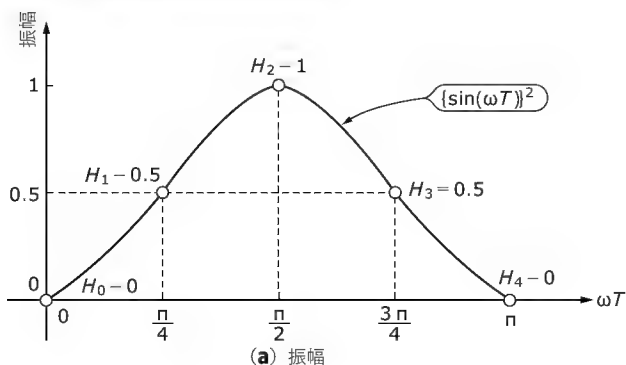
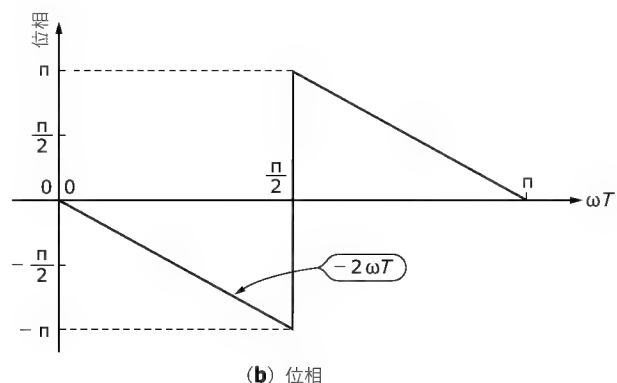
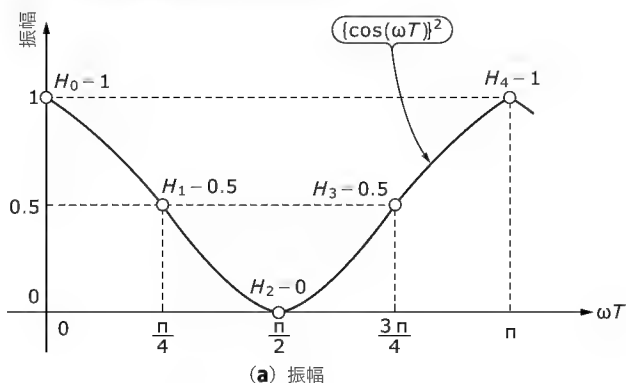
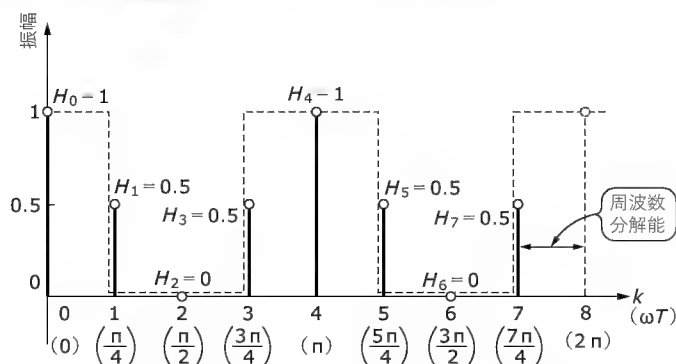
〔図 14.23〕 例題4 の周波数特性

(a) 振幅

(b) 位相

〔図 14.25〕 例題5 の周波数特性

(a) 振幅

(b) 位相

〔図 14.24〕 例題5 バンドストップフィルタの設計仕様

$$H_{BP}\left(e^{j\omega T}\right) = G\left(e^{j\omega T}\right) = \left\{\sin(\omega T)\right\}^2 e^{-j2\omega T} \cdots\cdots\cdots\cdots (41)$$

で求められる(**図 14.23**).

例題5

図 14.24 のバンドストップフィルタを設計し，周波数特性(振幅，位相)を示せ．

解答5

図 14.24 より，$H_0 = 1$，$H_1 = 0.5$，$H_2 = 0$，$H_3 = 0.5$，$H_4 = 1$，$H_5 = 0.5$，$H_6 = 0$，$H_7 = 0.5$ であり，$N = 8$ として式(4)の DFT 値は，

$$h_0 = 0.5,\ h_1 = 0,\ h_2 = 0.25,\ h_3 = 0$$
$$h_4 = 0,\ h_{-3} = 0,\ h_{-2} = 0.25,\ h_{-1} = 0 \cdots\cdots\cdots\cdots (42)$$

と計算される．よって，**図 14.25** のバンドストップフィルタの伝達関数を $H_{BS}(z)$ と表せば，$h_3 = h_4 = h_{-3} = 0$ を考慮して式(40)と同様の計算により，

$$H_{BS}(z) = (h_0 + h_1 z + h_2 z^2 + h_{-2} z^{-2} + h_{-1} z^{-1}) \times z^{-2}$$
$$= h_2 + h_1 z^{-1} + h_0 z^{-2} + h_{-1} z^{-3} + h_{-2} z^{-4} \quad \cdots\cdots (43)$$

で与えられ，周波数特性は，$z = e^{j\omega T}$ を代入して，

$$H_{BS}\left(e^{j\omega T}\right) = G\left(e^{j\omega T}\right) = \left\{\cos(\omega T)\right\}^2 e^{-j2\omega T} \cdots\cdots\cdots\cdots (44)$$

と求められる(**図 14.25**).

以上，例題2 ～ 例題5 に示すように，設計したいディジタルシステムの振幅特性を仕様として与えれば，FFT 計算により得られる DFT 値をフーリエ級数の展開係数とみなして伝達関数を導き出せることがわかる．つまり，所望の仕様を満たすディジタルシステムを設計できることが理解されるのである．

*　　　　　　*

次回は，引き続き“ システム設計編 II ”として，周波数サンプリング法とよばれる設計手法などについて解説する予定である．お楽しみに．

みたに・まさあき　東京電機大学工学部情報通信工学科

ハッカーの常識的見聞録㉖

今月の常識 デジカメとPC共用CFカードはバックアップを忘れずに

■ 広畑由紀夫

☆ディジタルカメラ(デジカメ)は，プロ用が1,000万画素を超え，一般用でも500万画素を超える高品位な製品が普及してきている．そのような状況では，PCとデジカメ間でメモリカードを共用することも多いだろう．今回は，CFカードにまつわる話をする．

コンパクトフラッシュ(CF)型のメモリカードストレージも1Gバイトマイクロドライブではなく，フラッシュメモリで1Gバイトの大容量のものが普及するようになり，早ければ2002年末には1.5G，2.0G，3.0Gバイト製品の発表/発売開始が話題にのぼってきています．また，アクセス速度もPIOモード4の速度(ATAインターフェース)で可能になる高速タイプのコントローラ搭載CFカードが一般用として発売されました．PIOモード4は，UDMA接続の最新ドライブに比べてたしかに転送速度だけを考えれば遅いものの，フラッシュメモリと従来のHDDを比べた場合，HDDでいう平均シーク速度はフラッシュメモリなら0に近くなります．

● フォーマットと再利用問題

CFカードをPCで簡単に使用できるアダプタやCFスロット対応のPocketPCや，Palmなどで CFカードをフォーマットして再利用することは，デジカメとPC間のデータ交換にCFカードでやりとりしている人には，もはやあたり前のことでしょう．最近では，デジカメとUSBなどで直接接続するキットも出始めてはいますが，やはりカードアダプタの値段の差は大きいもので，1,000円弱のアダプタカードと，1万円を越える接続キットのオプションのどちらを買うか，といわれると，アダプタを買って利用する人のほうが多いのではないかと思います．

さて，再利用してフォーマットする際，Windows XPでフォーマットすると，そのCFカードが使えなくなってしまうという現象が，巷でよく指摘されています．また，その際にはデジカメでのフォーマットさえできなくなってしまう機種も一部にはあるようです．

実際，筆者が先日購入した製品では，取扱説明書には「デジカメ本体でフォーマットしてください」とは書かれていましたが，「デジカメ本体以外でフォーマットしたカードは使用できません」とは書かれていませんでした．そこで，Windows XPでFATフォーマットを行った128Mバイトのカードをデジカメに入れてみたのですが，「このカードは使用できません」というメッセージが出るのみで，フォーマットすらできませんでした．メーカーのWebにあるFAQなどでも，「FATフォーマットにしてください」ということのみで，いまだに機種によってはPCでフォーマットしたカードが使用できないという例があるようです．

● 再利用に関する対処法

筆者の例では，デジカメ本体のメーカーサポート修理センターで対応をしてくれるということで一件落着はしましたが，中古や譲り受けたデジカメを使用する場合などには，メーカーのWebでフォーマットに関するFAQなどをよく調べておくべきでしょう．そのほかの情報としては，オリンパスから2002年の秋以降に販売されているデジカメで再フォーマットできるものが非常に多いようなので，メモリ購入店に展示機があれば頼んでみるのも手でしょう．

もっとよい手段は，やはりPCでなんとかできないかということですが，通常のフォーマットプログラムで対応できないのであれば，最後はフォーマット情報から再現できるバックアップ&リストアしかなくなります．

● CFカードのバックアップ

筆者の環境では，HDDドライブとしてのアダプタと，PCMCIAアダプタを使い分けています．そのほかにはUSBも使用していますが，こちらはおもに持ち歩き用です．

さて，今回検証してみた現象は，物理フォーマットには関連しておらず，論理フォーマットが異なっているために使用できなくなっているということが，フォーマットユーティリティとデジカメの動作から判断できます．そこで，論理フォーマットごとバックアップすればよいということになります．

こうした機能は，一般的なソフトウェアでは，DriveImageなどが発売されているので，それらを利用するとよいでしょう．そうしてドライブのイメージとしてバックアップファイルを作成しておき，同容量のメモリを使い回すときには，いったんリカバリを行うことで対応するというわけです．

● プログラムで行うなら……

Windows2000/XPなどで，I/Oデバイスコントロールによる，論理フォーマットのジオメトリ取得と，論理フォーマットのカスタムフォーマットプログラム，さらにデバイスが追加されたときに自動でバックアップを行うプラグ&バックアップツールをVisual Studio.NETで自作している最中です．こちらは機会をみて発表できるでしょう．

バックアップさえしておけば，ノートPCのHDDを大容量CFカードにして，高速起動ノートPCのような改造や使い分けも安心して行えるでしょう．

ひろはた・ゆきお OpenLab.

シニアエンジニアの技術草子 弐拾四之段

◆人間は考える葦である

旭　征佑

● 二宮金次郎

その石像は，小学校の校門を入ってすぐ左にあったので，学校に通うたびに毎日お目にかかっていた．母親から聞いた話だと，「貧しい中，忠義奉公しながらも，寸暇を惜しんで勉強した」お手本の存在だそうだ．石像は，重そうな薪を背負って働きながらも，しっかりと本を両手にもって勉強している姿をしている．古くなって苔が染みついている状況からも，母の話が真実味を帯びて聞こえた記憶がある．じつはこの石像には，筆者は特別な思い出がある．二宮金次郎を例に出し，毎日のように「勉強しなさい」と母親にさとされていたからだ．もちろん，今となっては，良い思い出でもある．

二宮金次郎の銅像や石像は，ほとんどすべての小学校にあったそうだ．しかし，今ではどこの小学校に行っても，ほとんど見かけることはない．文献も，探してみると意外と少ないことがわかった．しかしネットで検索すると，簡単に見つけることができる．それだけではない．「学校の怪談」にまで出てくることがわかって苦笑してしまった．そこでは，二宮金次郎が夜中に運動場や廊下を走っていたとか，翌日学校に来たら，石像が背負っている薪の本数が減っていたとか，そんな話らしい．わずかに石像が残っている小学校でも，得体の知れない像になり下がってしまっているのが，なんだか空しい．

ここで，二宮金次郎（二宮尊徳）を簡単に紹介しておこう．彼は，天明8年（1787年），農家に生まれた．14歳で父を，16歳で母を亡くし，傾いた一家を支えて一所懸命に働きながら，独学で読み書き，算術を身につけた．そして荒れた田畑を開墾し，得たお金で失った田畑を買い戻した．25歳のとき小田原藩の家老服部家に取り立てられ，経済的に疲弊していた服部家をわずか1年で再建，その後依頼を受けて多くの藩の財政も再建する．のちに五条講という名の新しい資金調達のしくみを考案し，最終的に全国1,000箇所以上に広めた．これは現在の信用組合，農業協同組合へと発展していく．

● 怠け者を増やす先人の知恵？

考えてみれば，先人たちはいろいろな知恵やしくみを残してくれた．おかげで我々は，それらを利用して便利に生活できる．インターネットも同じかもしれない．すべて先人の知恵のデータベースやしくみがネット上に蓄積されているからだ．

本などで調べても，専門外のことはひどく時間がかかったり，不正確だったりする．たとえば，二宮金次郎を調べようと思ったら，以前なら親に聞いたかもしれないが，正確な知識が得られるかどうか怪しいところだ．家に戻って，埃まみれの百科事典でもひっくり返すかもしれないが，型どおりの説明しか載っておらず，自分の知りたいところはなかったりする．Webを使って調べると，二宮金次郎の経歴や，石像・銅像の建てられた経緯，さらには学校の怪談に登場することまで，いろいろな方面の情報が載っており，その総数は数千件にものぼる．

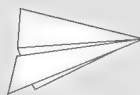
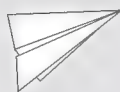
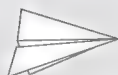
ほかにも，いままで多くの先人が蓄えてきたしくみそのものも，改良されてネット上に再現されてきている．おかげでネット上で買い物もでき，買った物は宅配便で送られてくる．ネット銀行も登場し，政府のe-Japan構想により官公庁の手続きも自宅や会社に居ながらにしてできるようになってくる．顧客獲得のため，当然ながら，同様のしくみを民間の企業も作るだろう．

しかし，いかにも知識と経験をすべて吐き出してしまったようにも見える中年おじさんが，インターネットの情報を単純に引用しているだけの状況に遭遇したりすると，非常に空しい．いったい何をやっているんだ．人間は考える葦ではなかったのか．

● 年を取ってもボケない

ただでさえ，人間は歳をとるとボケるといわれる．たしかに，簡単な漢字を思い出せなくなって，書いている途中に筆が止まったり，ちょっとしたことが思い出せずに，「あの，あれ」などと連発したりするようになる．筆者も若い頃はそんな四十すぎのおじさんを，思いっきり馬鹿にしていたものだ．おじさんたちが笑ってごまかしていたのは，一種諦めの境地に達していたのだろう．

そのボケというのは，基本的に記憶力の減退からくるものらしい．しかし，記憶をつかさどる器官は歳をとってもまったく衰退しないことが，最近の研究の結果わかったようだ．それにもかかわらず，なぜ歳を取るとボケるようになるのか？

歳を取ると，考えることや，記憶することをいつのまにか放棄していることに気がつく．このため，脳の機能が衰退していくのが現実のところだという．そういえば，80歳になっても90歳になっても，若い人に負けないくらい頭の切れる人はいる．つまり，年を取ったらボケるというのは誤りで，頭を使わなくな

ったからボケるというのが正しいのだ．

読者の方も振り返って考えてみるといい．最近，試験勉強や，何かを必死に暗記したことはあるだろうか．記憶をするというのは，脳のもっとも重要な作業だ．あらゆる面で，これを怠ってはいないか．

● 洗練された頭脳と強靭な足腰

しかし，心配は不要かもしれない．

最近は，大量のWebページを見るようになった．そして素早く要点を見つけ出し，整理するようになった．以前にくらべ，その情報量や処理量は圧倒的に増えている．もしかしたら，情報過多の時代，大量の情報の中から自分の求める情報を探し，素早く整理・分類し，処理してしまうために，脳はフル回転しだしているのではないだろうか．これは，情報が少なかった時代には不要だった，まったく新しい能力だ．短期間に大量の情報を記憶して，整理しなければならないのだから，人間は記憶力をフルに活用する．

こう考えると，インターネットの登場はボケを生むどころか，以前にもまして高度な頭脳を生み出してきているのではないかという考え方もできる．

たとえば，昼食を食べるときのことを考えよう．太古の時代，人間には，生きていくために，近くの獲物のありかを手当たりしだいに見つける動物的な勘があったはずだ．しかし現在，人間にはその能力はほとんどない．だが，近くのどんな店がランチをやっているか，どの店が何曜日に休みか，どの店がおいしいか，そんな多くの情報をもって，自分にあった店を即時に選択し，その店に向かうのが普通だ．おかげで，たった1時間の昼休みに外にでかけるだけで，昼食にありつくことができる．そして，栄養状態は昔よりすこぶる良好だ．つまるところ，人間はより多くの情報を得て整理することで，ますます進化してきたのである．

とはいえ，気になることが一つある．インターネットで物事が足りる分，歩き回る量が大幅に減っていないだろうかという点だ．以前，子供の玩具の万歩計を借りて，1日どのくらい歩いているか計ったことがある．以前は数千歩から1万歩以上も歩いていたこともあった．意外な多さに驚いたものだ．しかし，最近は多くて数千歩，ひどいと2千歩を切っている．歳のせいもあるだろうと思うが，歩き回る必要性が減ったことも大きく影響しているのではないかと思う．

椅子に座る時間が長くなると，どうも足腰を鍛えなくてはいけなくなるらしい．こればっかりは，どうしようもない．ほんのわずかなことだが，わざわざ机まで行ってちょっと打ち合わせするといったことまで，メールで済ましてしまうという事実もある．

パスカルが言った「人間は一本の葦にすぎない．自然の中でももっとも弱いものである．だが，それは考える葦である」．人間は考える葦かもしれないが，もう一つ，葦は立っている必要があることを忘れてはならない．足腰が弱り，折れてしまったら元も子もないのだ．

そういえば二宮金次郎も，多くの薪を背負い，山道を歩きながら本を読んでいたので，りっぱな足腰をもっていたことだろう．晩年になって，幕府の御普請役となった彼は，70歳で没するまで，次々と新しい施策を打ち出し，全国を歩いて農地開拓，経済改革を実現していくことができたのだ．

あさひ・しょうすけ　テクニカルライター
イラスト：森　祐子

起業・独立のステップ

H. Tony Chin

筆者のまわりにはさまざまな形で仕事をしている人が多い．フリーでエンジニアリングやその他のコンサルティングをしている人，会社を経営している人，そして大小さまざまな規模の企業に勤めている人達だ．今は会社に勤めていても，いずれは独立したいとか，起業してみたいという気持ちがある人達がいる．

日米に関わらず，企業にまだサラリーマンとして勤めている人達からよく聞く質問が，「どうやったらうまく独立・起業できるか？」である．勤めている会社を急に辞めてスタートアップを作ったり，独立するケースはまれだと思う．シリコンバレーでの起業のケースを見ても，余裕を見た準備期間をもって計画的に進めていることが多い．

☆ 何でもうけるか？

起業するなり独立するなりで，いちばん始めに皆が直面するのが「何をやってお金もうけをするか？」というポイントではないだろうか．さまざまな考え方があるが，やはり自分の今やっている仕事の延長線や現在の仕事に近いこと，またはその応用が現実的だろう．

たとえば筆者の知り合いで，その昔はRTOSの会社に勤めていたエンジニアがいる．8年ほど前に独立して，フリーでプログラミングの仕事をしていた．きっかけはサポートをしている客先が忙しいときに，プログラミングを手伝ってほしいというバイトの話があったという．サポートの領域をはるかに超えている内容だし，勤めている会社では副業を禁じられているので，当初は断ったそうだ．もっとも彼はRTOSの会社に勤めているからこそ自分の価値があると思っていて，会社を離れて一人で仕事をするとあまりメリットがないのではないかと自分を相当疑ったようだ．

しかし，同じような話がほかの企業からもあり，仲の良い顧客と話をする中で，自分のスキルは独立してもかなり使えるという感触を得たようだ．実際，独立してからは，前から付き合いのあった顧客から仕事を貰っている．顧客側も安心して仕事を出せる先があるし，彼としても安定して仕事が来るという構図になっている．

自分が普通にやっている仕事が，外から見ると意外に大事で価値があるというケースだ．また，辞めた会社からも仕事を紹介されるケースもあったそうだ．彼の場合は，謙虚なタイプだったので必要以上に自分を過小評価していたようだ．はじめの1年目は顧客からお金を取るところでだいぶ苦労したと言っていた．

さらにもう1人の知り合いは，プログラミングが3度の飯よりも好きで，いつも自宅に最新の開発環境をもっていたり，新しいプラットホームでのプログラミングに興味をもっている人だ．彼の場合は自分の立ち上げた会社の仕事以外に常に4～5個ネタを練っていて，暇さえあれば自分のアイデアを試すためのプロトタイプを作ったりしている．ジャンルはさまざまで，シミュレータからJavaで書いたWeb上アプリケーションまである．

彼は機会さえあれば，その道に詳しい人に自分のプロトタイプを見てもらったりして，ビジネスにするチャンスをうかがっている．実際にビジネスとしていくつか立ち上げているが，彼はネタ作りが好きなようで，製品化には向いていないと自分でも言っている．パートナを数名集めては立ち上げて，軌道に乗るとまた次のネタ作りをするようだ．ネタの中には，話にならないようなものもたくさんあるが，彼の場合は「数撃てば当たる」鉄砲的な考え方なのだろうか？　上場するような大きな会社にはまだ化けていないが，地道に製品を作っているともいえるし，自分の好きなことなので長続きしているのだと思う．

これらの二つの例でいえるのが「エンジニアはエンジニアらしい仕事で勝負」というところではないだろうか？　逆に新しいビジネスモデルや複雑なライセンス形態で起業をして，勝負に出た人がいる．かなり派手に投資家も集めた．しかし，お金のやり取りや，何をビジネスのネタにしているかが非常に理解しにくい会社であった．今から振り返ると，彼が起業したときはインターネットバブルのときなので，ある種の流行だったのかもしれない．今でも会社は続いているが，複雑なビジネスモデルのため苦戦しているのはたしかだ．

☆ 自分に関する棚おろし……正直に自分を評価してみる

前述の「何でもうけるか？」に関連して，自分に何ができるのかをはっきりと把握する必要がある．自分のもっている能力・スキルや得意分野，不得意分野などを，よく，正直に，的確に理解しておく必要があると思う．これによってビジネスの方向性を見つけたりすることができる．たとえば，エンジニアリング以外に自分は文章を書くのが得意だとか，説明をするのがうまいと社内でいわれている人は，ビジネスが軌道に乗るまで何らかの講師をするといった方法でマーケティング活動ができる．また，起業するなり独立するなり，社外の顧客や協力企業に説明をしたり，営業を行ったりしなければならないケースが出てくる．それまでにあまり社外に出る機会がなかったとか，知らない人と話をしたり説明するのが苦手な人もいると思う．不得意である分野は，これから起業・独立するまでに自分で身につけるなり，それを補ってくれるパートナを探すなどの準備が必要となる．会社に勤めて

エンジニアをしていると，どうしてもビジネス的なスキルに接する機会がないので不得意分野に入ってしまうのではないだろうか？　先ほどの例の営業や，一般的な経営のスキル，財務・経理のスキル，それから契約を取りまとめたりするスキルなどだ．一部は外部の専門家に頼めるものもあるが，できるだけ自分で理解を深めたりするのが大事なのはいうまでもないと思う．

経理，財務，営業や法律などの専門分野は，自分のやる気しだいで学んだり身につけたりすることが可能であると誰もがいう．そして，その姿勢で知識をつけていくのがシリコンバレーのエンジニア達の姿だ．しかし個人のネットワーク……ようするにコネは一夜で築き上げることはできないので，これだけは時間をかけるしかない．シリコンバレーでは，比較的横のつながりが多いため，社外のエンジニアや技術職でない人達と交流したり知り合いになれる機会がある．学会，展示会，会合など外に出る機会で多くの人と会い，交流を続けることでしか，ネットワークはできない．純粋にエンジニアリングの能力が評価されて会社がスタートするというケースはほとんどない．エンジニアリングができる人，ビジネス的なセンスのある人，そして投資をする人が引き合わされて話が進んでいくことがほとんどだ．やはり人間同士のやり取りであるから，顔つなぎができていて，お互いに信頼関係がすでに構築されているケースのほうが話が進みやすい．これには日米の差はあまりないと思う．どこの国でもやはり「良い出会い」はたいせつだ．

☆ 助走をつける……もっとも大事な準備段階

シリコンバレーのエンジニア達が起業・独立する場合，意外と知られていないのは，準備段階を万全にするというところだ．急に会社を辞めて，次の日からスタートアップ企業を設立した……という例はまずない．会社に勤めながら暇さえあればプロトタイプを作ってみたり，それを評価してくれる人を探すなど，期間を十分に取っている．多くの会社は副業をしたりすると問題になるので，だいぶ気をつかってこの期間を過ごすことになる．また，起業・独立に必要なスキルやコネ作りも十分に時間を取る．

筆者の知っている人達で起業・独立した人達は，「会社勤めを辞めて自分で何かしよう」といいはじめてから，準備期間を少なくても1年〜2年ほどもつ．プロトタイプの実験や勉強，情報収集ももちろんだが，スキル不足を克服するための自分の修行や，コネ作りにも十分に時間を費やす．また，個人的にも多くを犠牲にすることが多いスタートアップでは，家族の理解やその準備もたいせつだ．給料が急激に減るのは当たり前だし，貯金を起業のブートストラップに使うケースが多い．起業・独立でうまくいっている人達は，個人的に貯金をかなりもっていたり資産がある人が多い．個人的な財務管理が会社経営にも反映しているのだろう．また，違うスタートアップで一財産築いてから自分の会社を立ち上げる人も多い．いずれにせよ十分に時間と余裕を見ながら，自分の作りたい会社や製品を考えながら着々と準備をしていく．

シリコンバレーの場合だと，転職や社内異動でさまざまな経験を積むことが可能だ．プログラミングや開発のみをやっていたエンジニアが夜間でMBAを取り，あるときからマーケティングに配属されていたというケースもある．周囲もはじめは気付かないのだが，いままでジーンズで通っていたエンジニアが，ある日突然スーツで会社に来てびっくりされたという話もある．

もっとも起業のスキルを手っ取り早く身につけられるのは，スタートアップの社員として仕事をすることだろう．小さい会社だと，自然と役割の幅が広いうえにチャレンジをさせてくれる機会が多い．従業員が少ないが，やらなければならないことは山ほどあるからだ．だんだんとスキルアップをするごとにスタートアップのスケールを小さくしたり，設立年数が少ない会社を選んでいくと，気がついたら設立のメンバーの一人であったというケースもある．とにかく，まわりにさまざまなサイズのスタートアップ企業があるのでお手本には最適だ．

☆ お手本が多いメリット……ノウハウが集中している

スタートアップに必要なお手本となる人や会社が多いし，長年のテクノロジベンチャーを築き上げたノウハウや経験を積んだ人達が集中しているという意味では，シリコンバレーが有利なのはたしかだ．常に自分のキャリアを自分で管理する環境にあるシリコンバレーのエンジニアは，スタートアップの夢をもっている人なら少しずつでもそれに近づくための工夫があるようだ．自分の力や能力を的確に評価して足りない部分を補っていくプロセスを繰り返し，機会を見て，一度は自分の思ったとおりにやってみる人が多い．

トニー・チン　htchin@attglobal.net　WinHawk Consulting

Column

次の牽引役は？

シリコンバレーはインターネットバブル崩壊後の後遺症をまだ背負っている．貸しビルの空きがどんどん増えているし，一時期は困っていた日中の高速の渋滞もかなり緩和された気がする．全米の失業率が5.7％近くで，シリコンバレーでは7％近くまで上昇している．そして，まだレイオフのニュースは続いている．

一般的にアメリカ人は引っ越すことにあまり抵抗はないので，職や良い住処を求めて積極的に移動することが多い．だから，ダウンサイクルに入ってレイオフが当たり前になり始めると，かなりのペースでどんどん人口が移動していく．80年代，90年代にもダウンサイクルはあったが，その当時は，パソコンが駄目なら通信ネットワーク関係とか，モバイル系のハードが多く出荷されていたため，ある種の牽引役となったそうだ．今回のダウンサイクルでは，確実な牽引役がないところが問題のようだ．Y2Kでパソコンや IT関係のハード・ソフトの投資は十分にあるし，通信ネットワークのインフラも十分にあるからだ．軍事や政府関係のハードは，相当大きな戦争（朝鮮戦争やベトナム戦争ぐらい）にならないと牽引役にならないという話だ．やはり目玉となるハードがあり，それにさまざまな他の分野のテクノロジが利用されるわけだ．

HARD WARE

●32ビットマイコン

MB91F233 MB91232L

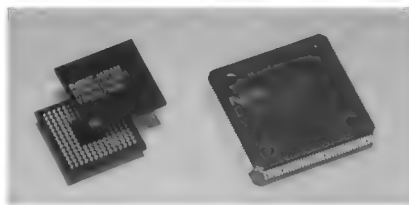
- 低消費電力化を目的として開発されたFR 60Liteシリーズ32ビットRISCマイコン.
- CMOS0.35μmプロセスを採用.
- 動作周波数は，2.1MHz～33.6MHz.
- 最大消費電流は，80mA(MB91F233)および50mA(MB91232L).
- 電源電圧は，内部(2.4～3.6V)およびI/O(3.0～5.5V)の2電源を装備.
- 搭載メモリは，256Kバイトフラッシュメモリ/16KバイトRAM(MB91F233)および192Kバイトフラッシュメモリ/16KバイトRAM(MB91232L).
- A-Dコンバータは，4チャネル×2ユニットの10ビット.
- D-Aコンバータは，2チャネルで8ビット.
- クロック出力機能，リアルタイムクロック機能，UART通信機能，各種タイマ機能を搭載.
- スリープ，ストップ，時計モードの各低消費電力モードを搭載.

■ 富士通(株)
サンプル価格：¥2,500(MB91F233/MB91232L)
TEL：042-532-1397
E-mail：edevice@fujitsu.com

●汎用1チップマイコン

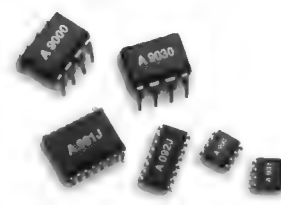
V850E/MA3

- 80MHz動作時に106MIPSの高速処理を実現する「V850E1」CPUコアを搭載しており，高性能なリアルタイム制御を必要とするアプリケーションに適する.
- 処理性能あたりの消費電力として，4.7mW/MIPS(Typ.)を実現.
- 1クロックで読み出し実行可能なROMを512Kバイト内蔵することで，高速なリアルタイム処理を実行可能.
- 1クロックアクセス可能なRAMを32Kバイト内蔵しており，スタック，ワーク用メモリ操作の高速化を実現.
- フラッシュメモリ内蔵製品では，ボードへ実装したままでプログラムを書き換えられる.

■ NECエレクトロニクス(株)
サンプル価格：¥900～¥3,600
TEL：044-435-9494　FAX：044-435-9608
E-mail：info@lsi.nec.co.jp

●アイソレータ

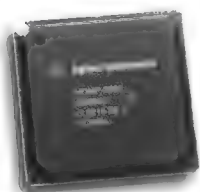
HCPL-9000/0900シリーズ デジタルアイソレータ

- 100MBd(Mbaud data)という伝送速度を，代表値で2nsのパルス幅歪み，および10nsの伝達遅延時間で実現.
- 2チャネル品で単一方向，双方向，4チャネル品で単一方向，2/2双方向，1/3双方向と多様な内部コンフィグレーションを用意.
- 製品パッケージは，8ピンDIPから8ピンおよび16ピンのスモールアウトラインパッケージまで用意.
- 3.3V供給電圧時の入力電流は1チャネルあたり10μA，供給電流は1出力チャネルあたり2mAの低消費電流を実現.
- 動作電圧は3.3Vと5.0V.
- 瞬時同相雑音除去電圧は，V_{cm}=1000V時に最小値15kV/μs.

■ アジレント・テクノロジー(株)
サンプル価格：¥500～
TEL：0120-61-1280

●シリアライザ/デシリアライザ

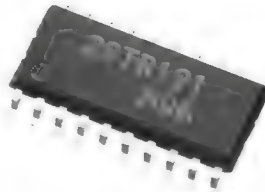
HDMP-2689 ファイバ・チャネルSerDes IC

- 19×19mmの289ピンPBGAのパッケージで供給される2Gp/sファイバチャネルの4チャネルSerDes.
- 2.125Gp/sと1.0625Gp/sの二つのシリアルデータ速度を4チャネル独立に設定することが可能.
- 10ビットのパラレルインターフェースに加え，8B/10Bエンコード/デコード機能を搭載.
- 0.18μmのCMOSプロセス技術の採用により，1.7Wの低消費電力を実現.
- IEEE802.3で規定されているMIIからSerDesのコントロールとステータス確認が行えるため，システムのダウンタイムを削減することが可能.

■ アジレント・テクノロジー(株)
サンプル価格：¥9,860
TEL：0120-61-1280

●モータ駆動用IC

形2STB121PM

- 疑似正弦波ソフトスイッチング方式を用いて，静音化を実現.
- 従来はファンモータで使用する，3相ブラシレスモータ駆動に必要だったホールセンサをなくすことで，モジュールの小型化，薄型化を実現.
- 正転，逆転切り替えの設定が可能.
- サーマルシャットダウン機能を装備.
- 1Wクラスのモータに対してセンサレス駆動を実現.
- 最大出力電流は12mA.

■ オムロン(株)
価格：オープン価格
TEL：075-344-7074

●液晶ディスプレイモジュール

NL6448BC33-53

- 広視野角，高輝度，高温度範囲をもつ，対角26cm(10.4型)アモルファスシリコンTFTカラー液晶ディスプレイモジュール.
- 見る角度によるコントラストや色味の変化を抑え，上下/左右，各170°の広視野角(コントラスト比10:1以上の範囲)を実現.
- パネル透過率を1.5倍に向上し，高輝度350cd/m^2，高コントラスト比300:1を実現.
- 動作温度範囲－10～＋70℃，保存温度範囲－20～80℃と広範囲な温度で使用可能.
- 表示色は262,144色で，画素数は640×480(307,200画素)を実現.
- インターフェースは，6ビットディジタルRGB輝度信号.

■ 日本電気(株)
サンプル価格：¥60,000
TEL：044-435-1851

●カラー液晶

D-TFD半透過型 カラーLCDモジュール

- ディジタルスチルカメラ向けクリスタルファインカラー液晶.
- 透過型液晶技術の採用により，外光環境に左右されず，表示の視認性が向上.
- 液晶の透過率を7%から10%に改善することにより，液晶表面輝度を180カンデラ/cm^2から250カンデラ/cm^2に向上.
- 透過率の向上により，小型モジュールの薄型バックライトの採用が可能となり，カメラの厚みをスリムにすることができる.
- 電池寿命を長くするために，バックライト消費電力を200mWから150mWに低減.
- 対角寸法は3.9cmで，表示ドット数は460×240ドット.
- 表示色は262,144色.

■ セイコーエプソン(株)
価格：下記へ問い合わせ
TEL：042-587-5816
URL：http://www.epsondevice.com/

●デュアルMOSFET

IRF7335D1

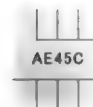
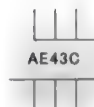
- nチャネル型MOSFETを2個とショットキダイオード1個を，14ピンSOICの1パッケージに収めたデュアルMOSFETであるFETKYデバイス.
- 二つのMOSFETは，それぞれ制御用と同期整流用に最適化され，同期整流用MOSFETと並列に内蔵したショットキダイオードは，順方向電圧が小さいため高速なスイッチングが可能.
- 同期整流用MOSFETとショットキダイオードとを接続する配線の浮遊インダクタンス，およびデバイスとプリント基板との接続の浮遊インダクタンスが最小限になるように設計されている.
- 耐圧はいずれも最大30V，ドレイン電流は最大10A.
- 制御用MOSFETのオン抵抗は13.4mΩ，ゲート電荷は13nC，同期整流用MOSFETのオン抵抗は9.6mΩ，ゲート電荷は18nC.

■ インターナショナル レクティファイアージャパン(株)
サンプル価格：¥250
TEL：03-3983-0875　FAX：03-3983-0642

●16ビットマイコン

AE45C M/Chip対応品 AE43C M/Chip対応品

- ICカード用アプリケーションである「Master Card M/Chip Select」を既存のICカード用16ビットマイコンのマスクROMに搭載.
- EEPROM領域の確保と迅速なICカードの製作，発行を実現.
- 16ビットマイコンのCPUコア「AE-4」を搭載し，EEPROM容量は32Kバイト(AE45C)，8Kバイト(AE43C)を内蔵.
- 暗号処理用のコプロセッサや電圧，周波数の異常検出器などの機能を備えているため，高セキュリティのICカードを実現.
- EEPROMの全容量をデータや他のアプリケーションプログラム用に使用することができるため，ICカードの機能を向上させるためのアプリケーションの追加などが可能.

■ (株)日立製作所
価格：¥620～¥720(AE45C/10,000個)
　　　¥500～¥600(AE43C/10,000個)
TEL：03-5201-5169
URL：http://www.hitachisemiconductor.com/jp/

●マシンビジョンカメラ

PC-640CL

- 30/60フレーム/sのCMOSイメージャ(640×480)搭載のカメラ.
- 最高1/10,000秒のフルフレームグローバルシャッタを搭載.
- 外部コントロールによる非同期リセット機能.
- カメラリンク，8ビットディジタル出力.
- RS-232-Cによるカメラコントロール.
- アドレッシブルスキャンにより，任意にフレームサイズをWOI(Window of interest，任意エリア抽出)操作で変更することが可能.
- 44×44×36.8mmの小型設計.
- フル作動での必要電力は，12Vで100mA.
- 900nmの波長域で50%の量子効率(入射光子あたりの出力電子数)は，近赤外感度にも優れる.

■ パルニックスアメリカ 日本支社
価格：下記へ問い合わせ
TEL：03-5805-2455　FAX：03-5805-8082

●パワーエレクトロニクス用ディジタル制御システム

特殊三相PWM出力ボード

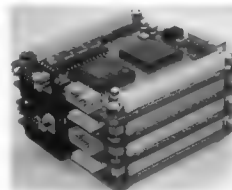
- 同社のパワーエレクトロニクス用ディジタル制御システムPE-Expert Ⅱの拡張ボードとして動作するPWMパターン出力ボード.
- 用途に合わせて各種三相PWM信号を生成可能.
- ゲート信号出力用光出力モジュールを搭載.
- 同じキャリア信号を使用し，同期した2または4系統の三相PWM信号を出力.
- 反転したキャリア信号を使用し，同期した2系統の三相PWM信号を出力.
- 中性点クランプ型3レベルインバータ用の三相PWM信号を出力.
- PWM出力中にキャリア周波数を連続的に変更することが可能.
- 外部信号に同期した三相PWM信号を出力.

■ マイウェイ技研(株)
価格：¥350,000～¥450,000
TEL：045-476-3722　FAX：045-476-3723
URL：http://www.myway-labs.co.jp/

●シミュレーション用ボードシステム

NPC-BCSシリーズ

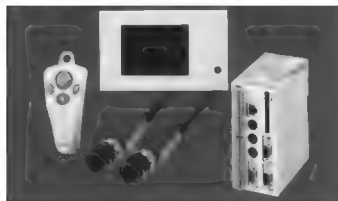
- マルチCPUシステムの開発環境を提供するボードシステム.
- 積み木状組み立て基板でマルチCPUシステムを実現.
- 二重化CPUシステム用のコンパクトな物理構造.
- 全基板の論理回路にはPLDを採用.
- VMEバスの基本機能を拡張した共通バスの採用.
- 共通バスは物理的，電気的な変更，データアドレス幅の拡大で高機能仕様を実現.
- 基板構造はマルチCPUシステムに必要な物理的ディジーチェーンを提供し，高速アービトレーション機能を実現.

■ (株)ナパック
価格：¥120,000
TEL：042-763-7737　FAX：042-763-7737
URL：http://members.jcom.home.ne.jp/napac/

HARD WARE

●画像処理装置

AX30シリーズ

- 外観形状認識を行うマイクロイメージチェッカのカラータイプ.
- 設定，操作の流れを重視したメニュー構成.
- 同社Aシリーズ共通のキーパッド.
- 抽出したい色をポインタで選択するだけで，色抽出が可能.
- 前面集中配線により，メンテナンス性を向上した.
- 液晶モニタバックライト50,000時間保証.
- 本体とモニタの着脱が可能.
- フルカラー液晶VGAモニタ対応.
- プルダウンメニュー，メッセージエリアを設置.
- 最大512Mバイトのコンパクトフラッシュに対応.

■ 松下電工（株）
価格：オープン価格
TEL：06-6906-1850

●ダウンコンバーティングミキサ

LT5512

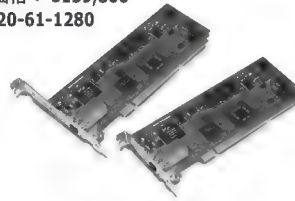
- パッシブミキサに対する高直線性の代替デバイスで，変換利得が得られ，LO（Local Oscillator）ドライブレベルを大幅に引き下げることが可能.
- 差動LOバッファアンプとアクティブ二重平衡ミキサを内蔵.
- 内蔵のRFバッファアンプによってLO-RF絶縁が改善されており，高精度の外付けバイアス抵抗が不要.
- 温度が補償されたバイアス回路とパワーダウンイネーブル/ディセーブル回路を内蔵.
- 950MHzで＋20dBm，1900MHzで＋17dBmの高入力IP3（3次ひずみ）を実現.
- 1900MHzで14dBのSSBノイズフィギュア.
- 4×4mmQFNパッケージに収容され，4.5V～5.25Vの電源電圧範囲で動作.

■ リニアテクノロジー（株）
サンプル価格：¥445
TEL：03-5226-7291
FAX：03-5226-0268

●iSCSI準拠PCIアダプタボード

ANIC-2101A/2103A

- IETFのiSCSI規格に準拠しているため，ストレージ装置の安定性や相互接続性を高めることが可能.
- TCP/IPの処理をハードウェア的に行うことで，CPUやOS側の負荷を軽減できるTOE（TCP/IP offload Engine）技術を組み込んでいる.
- アダプタボードのデータ処理能力を高めながら，システム全体の負荷やネットワークのレイテンシを削減することが可能.
- iSCSIレベルおよびTCP/IPレベルでアダプタボードにアクセスできるため，ブロック転送およびファイル転送の双方に対応可能.
- サーバに対してはターゲットとして，バックアップ機器に対してはイニシエータとして，処理を同時に行うことが可能.

■ アジレント・テクノロジー（株）
サンプル価格：¥139,800
TEL：0120-61-1280

●マルチメディアカード

HB28H016RM2/HB28D032RM2/HB28B064RM2

- マルチメディアカードの約半分のサイズのリデューストサイズマルチメディアカード（RS-MMC）.
- サイズ18×24×1.4mm，重さ0.8gの小型，軽量化を実現.
- 機能，ピン数，薄さは標準サイズのマルチメディアカードと同じで，マルチメディアカードとの互換性を維持している.
- 書き込み速度は1.0Mバイト/sを実現.
- 読み出し時電流28mA（Typ.），書き込み時電流33mA（Typ.）の低消費電力を実現しており，モバイル機器のバッテリ長時間駆動に適する.
- HB28H016RM2は16Mバイト，HB28D032RM2は32Mバイト，HB28B064RM2は64Mバイトの記憶容量をもつ.

■（株）日立製作所
価格：オープン価格
TEL：03-5201-5021

●CPU評価ボード

[ju:]シリーズ PXA250 EVA BOARD

- インテル製PXA250アプリケーションプロセッサの機能および性能評価，ハードウェアのリファレンスボードとして利用可能なシステム.
- G4250EB-X001（標準モデル），G4250EB-X002-S（20キー搭載モデル），G4250EB-X002-L（20キー，カラーLCD搭載モデル）の3タイプを用意.
- CPUの信号線を接続したコネクタを実装し，外部に拡張したオリジナル基板の作成が可能.
- バスクロック100MHz.
- SDRAM 64Mバイト，フラッシュメモリ32Mバイト，RS-232-C×1ポート，Ethernet×1ポート，7セグメントLED×8を搭載している.

■（株）ゼネティック
価格：¥148,000～¥228,000
TEL：03-3357-3044　FAX：03-3354-6144
URL：http://www.genetec.co.jp/

●無線LANシステム

UNI-LINK2410

- 2.4GHz帯の電波を利用した，無線間データ伝送速度100Mbpsをもつ高速無線LANシステム.
- 1台の親機で最大127台の子機の管理が可能.
- 電波妨害による通信不能の発生を避けるために，データ変調方式として使用周波数帯域がランダムに変化する，OFDM-PH複合方式を採用.
- OFDM変調方式におけるサブキャリアの数を増やすことによって，無線間データ伝送速度100Mbpsを取得.
- 無線間データ伝送において，ディジタル多重無線データ伝送方式の実現が可能.
- 無線の傍受によるデータの漏洩を防ぐため，システム間セキュリティのほか，リアルタイムにおけるDES-3暗号処理方式を上回る機能をもつ，独自方式による動的暗号処理を実装.
- ネットワークに対してブリッジ機能をもつ子機と，ルータ機能をもつ親機を用意.

■ ユニパルス（株）
価格：下記へ問い合わせ
TEL：03-5148-3866　FAX：03-5148-3001
URL：http://www.unipulse.com/jp/

●アナログストレージオシロスコープ

TS-80600

- 高輝度，可変残光機能付き，最高目視ライティングスピード10div/nsを実現．
- ストレージ機能により，高速単発現象をストレージ可能．
- DC～600MHz(50Ω)の高帯域で，4チャネルを搭載．
- 800×480ドットの高精細カラーディスプレイを搭載．
- 波形の重ね書きができる，パーシスタンス機能を搭載．
- プリンタ内蔵，LANインターフェース搭載，PCMCIAメモリカードの利用が可能．
- NTSC/RGB映像出力端子を標準で装備．

■ 岩通計測(株)
価格：¥1,480,000
TEL：03-5370-5474　FAX：03-5370-5492
E-mail：info-tme@iwatsu.co.jp

●マイナスイオン測定器

COM-3100

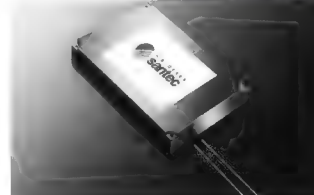
- マイナスイオン環境の測定，記録，データ化が可能な，鉱石用マイナスイオン測定器．
- イオン測定，連続イオン測定，ガイガーカウンタ積算の3種類の測定方式を用意．
- 最大測定値を999,999カウントまでサポートし，16桁2行のSTN液晶表示器を採用．
- リアルタイムデータ，メモリデータのプリンタ出力が可能．
- データをリアルタイムでRS-232-Cに出力でき，メモリデータをExcelフォーマットでパソコンに出力可能．
- AC100Vと内蔵電源の2電源方式で，1回の充電で約10時間使用可能．

■ コムシステム(株)
価格：¥398,000
TEL：042-543-9062　FAX：042-543-9570
E-mail：com@com-system.co.jp

●ラムダブロッカー

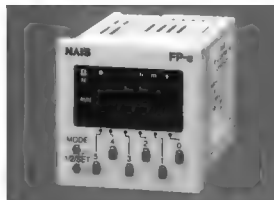
LBM-8

- ROADM(波長選択可能な光アッドドロップ装置)で用いるモジュール．
- 最新の光MEMS(Micro Electro Mechanical Systems)技術を利用した光可変減衰器を採用し，コンパクトな外形が実現され高密度実装が可能．
- 4チャネル単位の増設が可能であるため，初期投資の軽減を実現．
- 通過波長に同社の高性能干渉膜フィルタ技術を利用した，4スキップ0タイプの100GHz間隔WDMフィルタを搭載することも可能で，低損失なROADMの構築が可能．
- 温度特性に優れた多層膜干渉フィルタを使用しており，温度制御が不要．
- 10μW/チャネル以下の低消費電力を実現．

■ サンテック(株)
価格：¥400,000
TEL：0568-79-3536　FAX：0568-79-1718

●プログラマブルコントローラ

FP-e

- 盤面取り付け可能なプログラマブルコントローラで，サイズは48×48×77.5mm，PLC機能を内蔵．
- 操作盤に付いていたタイマやカウンタが不要になるため，操作盤の小型化，制御盤の省スペース化や制御盤レスも可能．
- 3色，2段5桁表示で，簡易アルファベットや数値を表示できるため，簡単なメッセージやタイマカウンタなどの設定値，経過値の表示が可能．
- 操作用スイッチで表示している設定値の変更が可能．また，入力スイッチとして使用できる．

■ 松下電工(株)
価格：¥29,800
TEL：06-6908-1131

●開発支援ソリューション

MJX330 for ARM

- PCカードTYPE II型のJTAGエミュレータ．
- ノートパソコンのカードスロットに挿入してターゲットにケーブルを接続するだけで，デバッグ環境の構築が可能．
- ターゲットシステム上のJTAGコネクタに接続するだけで，透過性に優れた安定したデバッグを実現．
- ARMコアに含まれる組み込みICE-RT機能をフルに活用することで，命令実行ブレーク，アクセスブレークなどのフルICEに匹敵するデバッグ機能をサポート．
- 標準添付のアセンブラレベルデバッガMJXDEBWに加え，グリーンヒルズ社製Multiにより，高級言語デバッグ環境を構築することが可能．
- オンチップデバッグ機能を利用し，プロセッサを実装した状態で安定したデバッグが行える．

■ (株)ライトウェル
価格：下記へ問い合わせ
TEL：03-3392-3331　FAX：03-3393-3878
E-mail：ZAXSales@lightwell.co.jp

●DSP向けスタータキット

TMS320C5510 DSK

- 200MHzで最大400MIPSの処理性能をもつ「TMS320C55x」を搭載した開発ボード，エミュレータ，コネクタ，統合開発環境「Code Composer Studio」，パワーマネジメント機能で構成．
- 統合開発環境「Code Composer Studio」内に，ソフトウェアベースのパワーアナライザとパワースケーリングライブラリを内蔵．
- パワーアナライザの使用により，コアレベルまたはシステムレベルの消費電力を視覚的に測定，分析し，消費電力の配分の正確な測定が可能．
- パワースケーリングライブラリは，ランタイムにおけるCPU周波数とコア電圧を計測し，周波数と電圧の最適な組み合わせを実現．
- パワースケーリングライブラリとともに，独立したコンフィグレーションライブラリを供給．これによりパワースケーリングライブラリをカスタムハードウェアやコンフィグレーションライブラリのソースファイルに移行することが可能となる．

■ 日本テキサス・インスツルメンツ(株)
価格：¥49,800
FAX：0120-81-0036
URL：http://www.tij.co.jp/pic/

SOFT WARE

●統合プラットホーム

Unify NXJ

- J2EE アプリケーションのライフサイクルを自動化，合理化する統合化されたアプリケーション環境を提供.
- RAD ツールで統合されたコンポーネントフレームワークにより約70%の作業を自動化.
- イベントドリブンのアプリケーションモデルに基づいており，アプリケーションの各イベントセクションに対しコントロール，プロパティ，アクションなどの定義が可能.
- アプリケーションの挙動とビジネスルールを J2EE BluePoint と融合し，Java の知識なしでコード生成，ビジネスルールの適用，異種データソースの統合が可能.

■ ユニファイジャパン (株)
価格：¥600,000 (開発版)　¥960,000 (実行版)
TEL：03-5614-5367　FAX：03-5614-5368
E-mail：contact@unify-jp.com
URL：http://www.unify-jp.com/

●Java 開発ソリューション

JBuilder 8

- JDK1.3.1 と比較して 20～80%のパフォーマンス向上を実現した JDK1.4 上で稼動させることで，開発パフォーマンスの大幅な向上を実現.
- Web アプリケーションの構築に対応する最新のオープンソースフレームワーク Apache Struts をサポート.
- 2Way ビジュアル EJB デザイナによる EJB 開発，複雑なエンティティ Bean のリレーションやデータマッピングの定義，XML 配布ディスクリプタのビジュアル定義など，EJB 1.1/2.0 準拠のアプリケーション構築を大幅に簡素化.
- SOAP，WSDL，UDDI，WSIL を含む最新の Web サービステクノロジをサポート.
- プロジェクト内のソースコードは，UML モデル化して視覚的に把握でき，マウス操作によってダイヤグラム間を移動することで，未知のコードの構造や関連，依存関係の把握が可能.

■ ボーランド (株)
価格：¥48,000 (SE 版) ¥360,000 (Enterprise 版)
¥520,000 (Enterprise Performance Bundle 版)
TEL：03-5350-9380　FAX：03-5350-9369
URL：http://www.borland.co.jp/

●組み込み用評価キット

プロトタイプキット

- 開発テーマごとに，組み込みソフトウェアの新技術の評価，実験，検証が可能.
- 技術テーマごとに必要となる組み込み用ソフトウェア部品をあらかじめリアルタイム OS に移植し，コンパイラ，デバッガなど開発ツールへの対応を済ませ，CPU ボードと適当な I/O から構成されるハードウェアプラットホーム上に構築.
- 組み込み無線 LAN + IPv6，組み込み無線 LAN，ユビキタス Bluetooth-LAN，Bluetooth マルチシリアルポート，ユビキタス IrDA，ホームゲートウェイ，ユビキタス IP ネット，ネット&ファイル，組み込み Web サーバ，IPv6/IPSec，クイック LAN，インターネット端末，USB2.0，簡単 GUI，メモリースティック，SD/SDIO メモリ，フラッシュファイル，リアルタイム Linux，リアルタイム BSD，オープンソース μITRON，メモリ保護 ITRON，組み込み Linux の 22 種類のプロトタイプキットをリリース.

■ (株) エーアイコーポレーション
価格：下記へ問い合わせ
TEL：03-3493-7981　FAX：03-3493-7993
E-mail：sales@aicp.co.jp
URL：http://www.aicp.co.jp/

●USB OTG プロトコルスタック

GR-USB/OTG

- USB 2.0 On-The-Go 仕様準拠のプロトコルスタック.
- USB 2.0 フル/ロースピードに対応.
- SRP，HNP をサポート.
- OTG 仕様の Descriptor，b_hnp_enable，a_hnp_support，a_alt_hnp_support をサポート.
- CPU に依存しない.
- μITRON をはじめとする各種リアルタイム OS に対応.
- 主要 OTG コントローラに対応.
- プラグ&プレイのための API 関数を提供.
- バルク/コントロール/インタラプト/アイソクロナス転送をサポート.
- さまざまなクラスドライバを提供.

■ (株) グレープシステム
価格：下記へ問い合わせ
TEL：045-222-3761　FAX：045-222-3759

●組み込みソフトウェア開発支援ツール

T-Navi

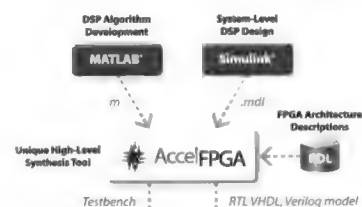
- 組み込みの対象となるハードウェアのインターフェースをソフトウェアとして提供する，開発支援ツール.
- レジスタ Read/Writeからの遅延時間設定による割り込み，および割り込み発生ボタンによる任意のタイミングでの割り込みの 2 種類を用意.
- デバイスが格納するレジスタ情報の定義，ソフトウェアのレジスタ Write 情報の表示など I/O レジスタのエミュレーションをサポート.
- レジスタ設定，割り込みデバッグ用データの保存やカーネル状態のトレースログの保存.
- タスク状態とシステムコール，レジスタ Read/Write，割り込みなどのタイムチャートの表示が可能.
- 固有の μITRON 仕様と TOPPERS/JSP カーネルとの差異を補間するインターフェース機能をもつ.
- 各半導体種別固有の拡張システムコールや構成情報の自動生成機能などを提供.

■ (株) 日立システムアンドサービス
価格：¥360,000 (1 ライセンス)
～¥2,880,000 (20 ライセンス)
TEL：052-263-3912　FAX：052-263-1162

●FPGA 動作合成ツール

AccelFPGA

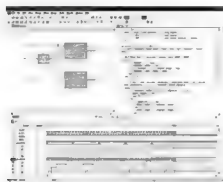
- 米国 AccelChip 社が開発した，DSP 向けシステムレベル設計ツール「MATLAB」，「Simulink」の出力ファイルから，FPGA 向けの HDL を動作合成するツール.
- 生成された HDL コードは，各社の FPGA 用論理合成ツールへの入力が可能.
- ターゲットデバイス向けに最適な HDL を出力する「TARGET」，データパスレジスタ挿入のための「PIPELINE」，処理リソース並列化のための「UNROLL」などの最適化指示を用意.
- MATLAB モデルのコンパイル後に，スケジューリング，リソース使用，パイプラインなどのレポートを得ることが可能.

■ 丸文 (株)
価格：¥6,000,000～
TEL：03-3639-5301　FAX：03-3639-2358

SOFT WARE

●多次元設計ツール

nVisage

- PCB 設計および FPGA 設計のための階層的回路図入力が可能な設計ツール.
- OrCAD の回路図，あるいはライブラリとの双方向の互換をもつ.
- ネストされたマルチチャネル回路図設計をサポート.
- 回路図と VHDL の混合設計.
- 複数の FPGA アーキテクチャに対応した VHDL ドリブン RTL 合成.
- SPICEf5/X-Spice 準拠のシミュレーションが回路図から起動可能.
- タイミングバックアノテーションをサポートする VHDL 合成.

■ アルティウムジャパン（株）
価格：¥348,000
TEL：03-5436-2501　FAX：03-5436-2505
E-mail：info@altium.co.jp

●組み込み向け USB プロトコルスタック

AVE-USB

- セイコーエプソン製 OTG 対応 USB コントローラ S1R72005 に対応.
- USB OTG 対応により，パソコンを介さず機器同士の接続を実現.
- ダイナミックにホスト/ペリフェラルの切り替えが可能.
- 必要な機能のみを選択して搭載でき，メモリ量を削減可能.
- プラグ＆プレイに対応.
- OTG 対応を意識して設計，開発.
- ANSI 仕様の C 言語で記述され，さまざまな CPU，OS に移植可能.
- 組み込み機器への最適化により，コンパクト化，高速化を実現.
- オブジェクト交換プロトコル OBEX をオプションで提供.

■（株）ACCESS
価格：下記へ問い合わせ
TEL：03-5259-3685　FAX：03-3233-0222

●アプリケーション移植ツール

Visual MainWin

- Windows アプリケーションを UNIX 環境に移植する開発ツール.
- Windows 用ソースコードを UNIX コンパイラにより再コンパイルし，ネイティブ UNIX アプリケーションを生成.
- インターネットインフラストラクチャ開発者やアプリケーション開発者が Windows 上でアプリケーションを開発し，それと同時に Windows 用と UNIX 用のモジュールの生成が可能.
- エミュレータとは異なり，開発から配布までをカバーするコンプリートクロスプラットホームソリューション.

■ エクセルソフト（株）
価格：オープン価格
TEL：03-5440-7875　FAX：03-5440-7876
E-mail：xlsoftkk@xlsoft.com
URL：http://www.xlsoft.com/

●組み込み開発プラットホーム

MontaVista Linux Professional Edition 3.0

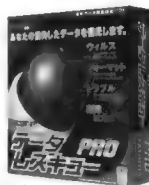
- サポートするすべてのアーキテクチャに対し，あらかじめビルドしてあるバイナリおよび統合されたソースコードを提供.
- カーネルは Linux2.4.18 をベースとしており，開発環境は豊富なアプリケーションパッケージがサポートする GNU ツールの最新バージョン，GDB5.2 と GCC3.2 を採用.
- メモリマネジメントの強化，IPv6 のサポート，さらに幅広いアーキテクチャをサポートしたシステムおよびユーザーレベルのイベントトレーシング機能，ジャーナリングファイルシステム，XFS などが含まれる.
- 広範囲な組み込みプロセッサアーキテクチャ，CPU ボードおよびソフトウェアコンポーネントをサポートした包括的な組み込み開発パッケージ.
- 七つの主要な CPU アーキテクチャから 24 種類以上のプロセッサ，70 種類以上のボードをサポート.
- クロス開発機能に焦点を当てており，11 種類のホスト環境で開発ツールの使用が可能.

■ モンタビスタソフトウエアジャパン（株）
価格：下記へ問い合わせ
TEL：03-5469-8840

●消失データ復元ソフト

R-Studio データレスキュー
R-Studio データレスキュー Pro

- パソコンの消失したデータを復元できるソフトウェア.
- 操作画面はエクスプローラ形式で表示され，消失データファイルを一目で把握.
- 「ログウィンドウ」には操作の履歴が「メインパネル」にはドライブのパーティション/FAT などのデータが詳細に表示.
- 復元させたいファイルだけではなく，複雑に構成されたフォルダでも，ツリー構造を保ったまま復元することが可能.
- 復元したデータは，ホスト OS で認識されるすべてのディスクに保存可能.
- ハードディスクだけではなく，FD や MO，デジカメのメモリカードなど，ドライブとして認識される媒体であれば復元可能.
- 消去されて間もないデータを検索する「ファイルサーチ」機能を搭載.

■（株）住友金属システムソリューションズ
価格：¥9,800
　　　¥15,800（Pro 版）
TEL：03-5815-7258
E-mail：dr-info@smisol.co.jp
URL：http://www.smisoft.com/

●ソフトウェア開発向けツール

Tau Generation2

- リアルタイムシステム開発向けツール.
- UML2.0 に対応したビジュアル設計であるため，正確で詳細な設計が可能.
- システムアーキテクチャの仕様だけでなく，システムの動作もビジュアルに明記.
- UML2.0 で提案されている実行形式のモデルに対応し，自動コードの生成が可能.
- モデルのシミュレーションにより，すべての開発工程で進捗を検証でき，早期のエラー発見が可能.
- ツールに制限されることなく拡張でき，モデルシステムの大きさに依存しないスケーラビリティを実現.

■ 日本テレロジック（株）
価格：下記へ問い合わせ
TEL：03-6402-1620　FAX：03-6402-1621
E-mail：info@telelogic.co.jp

IPパケットの隙間から

物持ちが良い人々の話

祐安重夫

友人のS君からメールがきて，32Mバイト分のSIMMが必要だけれど，たしか壊れたDELLのマシンがあったのではないかといってきた．たしかにマシンはあるが，壊れてはいない．それどころか，まだ現役で生きている．

このマシンは初期型PentiumのWindows95システムで，現在でもVMware上のWindows98SEから，リモートCD-ROMのアクセスに使用している．VMwareからは直接，ホストであるLinuxマシンのCD-ROMはもちろんアクセスできるが，Linuxで使用することがけっこうあるので，常駐させておきたい辞書CD-ROMは，古いマシンのほうに入れたままになっている．

VMware上のWindows98SEには，仮想CD-ROMドライバもインストールしてあるのだが，ホストとなっているLinuxマシンのメモリが256Mバイトしかないため，VMware仮想マシンにはあまりメモリを割り当てられない．そのため，仮想CD-ROMドライバは使用するメモリがもったいないのと，10Mbpsで接続されたリモートCD-ROMでも，辞書程度なら十分なスピードがあるため，こちらを使用することが多い．また，古いマシンには，昔のバージョンのWebブラウザなどが入っているので，ユーザーサポートの際などに役にたつことがある．

S君は，それではということで，さらに古い44MHzの486マシンはどうかと言い出した．これは7年前に買ったもので，S君のところも同じマシンを入れ，彼の会社も筆者の会社もしばらくの間，それにBSD系のUNIXを入れてサーバとして使用していた．このマシンも，16MバイトのSIMMを使用していた．S君のところではだいぶ前にサーバをリプレースし，筆者のところでは数年前までこれをサーバとして使用していた．しかし，S君のところと違い，うちではサーバのリプレース後もバックアップ用としてそのまま稼働させ続け，現在でも現役である．最近ではCPU負荷が高くて長時間動作するプログラムを走らせたり，ちょっとした並列処理プログラムの実験に，未だ使用している．S君には「物持ちがいい」と言われてしまったが，今になってSIMMを探すS君のほうも，けっこう物持ちが良い．

実際のところ，WindowsはさておきUNIX系のOSの場合，GUIにGNOMEのような負荷の高いシステムを使用しないかぎり，X+fvwm2程度なら44MHzの486マシンでも，それほどのストレスなく動作する．ましてXさえ使用しなければ(実際にはLinuxマシンからリモートでktermを立ち上げて使用しているが，もちろんローカルにXを立ち上げるより負荷は低い)，十分実用になる．その点では中古のマシンをたくさん買って，実験的にクラスタを構成する程度なら，かなり安価に実現できる．ただ問題は，場所をとるという点である．

ところで，知人から頼まれて中古のノートパソコンを探していた．なんでも冬は寒いから，こたつでパソコンを使用したいというのである．こたつでノートパソコンだと，結露が心配である．ずっと昔，北海道でこたつでApple IIを使用していたところ，結露で故障してしまったという話があった．まだアップルの日本法人がない頃で，アメリカの技術者にどうしても状況を理解してもらえなかったらしい．

それはさておき，インターネット通販のカタログをチェックしていたら，RedHat Linux 6.2をインストール済みのノートが14,000円で売られている．ただし，NICがついていないので，LANに接続するわけにはいかない．PCカードのNICを接続すると，本体より高くなりそうなのが難点だが，この価格なら使い捨てで使用しても惜しくはない．ちなみにRedHat 7.2をインストールしたものも5万円近くであったが，これもNICなしである．こちらはちょっと，使い捨てには惜しい．

どちらにしても，場所を取らずにLinuxでクラスタを組むことは，そんなに高額の投資を必要とはしない．もっともクラスタの本来の目的が，信頼性の向上や負荷の分散であることを考えると，あくまでも実験レベルの話ではあるが．しかし，簡単に実験できる環境を自前で用意することができるというのは，いろいろな点で便利だろう．

物持ちが良いといえば，MS-DOS時代の16ビットのパソコンは，手元にあったものはほぼ全滅してしまったが，20年近く前のZ80のCP/Mマシンが，まだ現役で生き残っている．さすがにLANには接続できないが，RS-232-CでBSDサーバに接続されている．

BSDサーバには1ボードで8ポートあるRS-232-Cボードが入っており，この時代にモデムが2台，MIDI音源が3台つながっている．さらにLinuxマシンとも，LANとは別に1系統シリアルで接続されており，これはおもにBSDサーバに接続されたMIDI音源に，LinuxやVMware上のWindows98SEからシリアルMIDIで信号を送出して，BSDサーバ側でMIDI機器に中継するといった目的で使用している．

CP/MマシンもRS-232-Cポートに接続されているわけだが，まだ3ポートも余っている．もう何台かMIDI音源を接続しようかとも思ったが，最近の音源はUSBが主流である．RS-232-Cの使い道も減ってきた．実際にCP/Mマシンに関しては，ファイル転送，リモートログイン，リモート実行といった機能が使用できるようになっているが，これらのソフトウェアの一部は自作で，9年ほど前の本誌でそれについて書いたことがある．

すけやす・しげお インターメディア・アクセス

海外・国内イベント/セミナー情報

INFORMATION

海外イベント

2003/1/4-8 **International Conference on VLSI Design**
Taj Palace, Nel Delhi, India
International Conference & Exhibtion Services
http://vlsi.ccrl.nj.nec.com/~chak/vlsi2003/index.html

1/6-10 **Macworld Conference & Expo**
The Moscone Center, SA, USA
IDG
http://www.macworldexpo.com/macworld2003/V33/index.cvn

1/9-12 **2003 International CES**
Las Vegas Convention Center, NV, USA
Consumer Electronics Association
http://www.cesweb.org/

1/14-16 **COMDEX SCANDINAVIA**
Swedish Exhibition and Congress Centre, Goteborg, Sweden
Key3Media
http://www.key3media.com/international/events/index.php?d=scandinavia&s=welcome

1/27-30 **COMNET Conference & Expo**
Washington Convention Center, WA, USA
IDG
http://www.comnetexpo.com/comnetexpo/V33/index.cvn

2/4-6 **Digital Content Delivery Expo**
San Jose Convention Center, San Josa, CA, USA
PBI Media
http://www.replitech.com/

2/9-13 **International Solid-State Circuits Conference**
San Francisco Marriott Hotel, San Francisco, CA, USA
IEEE
http://www.isscc.org/isscc/

国内イベント

1/19-23 **International Micro Electro Mechanical Systems Conference**
国立京都国際会館(京都府京都市)
IEEE
http://mems.kaist.ac.kr/mems2003/

1/21-24 **ASP-DAC 2003(Asia and South Pacific Design Automation Conference 2003)**
北九州国際会議場(福岡県北九州市)
日本エレクトロニクスショー協会
http://www.aspdac.com/indexj.html

1/22-24 **電子コンポーネント EXPO**
東京国際展示場(東京ビッグサイト, 東京都江東区)
リードエグジビションジャパン
http://web.reedexpo.co.jp/ed/

1/30-31 **Electronic Design and Solution Fair 2003**
パシフィコ横浜(神奈川県横浜市)
日本エレクトロニクスショー協会
http://www.edsfair.com/frame.html

2/5-7 **NET&COM 2003**
日本コンベンションセンター(幕張メッセ, 千葉県千葉市)
日経BP社
http://expo.nikkeibp.co.jp/netcom/exhibit2003/

2/12-14 **ISS Japan 2003(Industry Strategy Symposium Japan)**
パンパシフィックホテル横浜(神奈川県横浜市)
SEMI
http://www.semi.org/web/japan/wexpositions.nsf/url/iss03_j

2/26-28 **IP.net JAPAN 2003**
東京国際展示場(東京ビッグサイト, 東京都江東区)
リックテレコム
http://www.ric.co.jp/expo/ip2003/index.html

開催日, イベント名, 開催地, 問い合わせ先の順

セミナー情報

第1回組込みソフトウェアに関する教育・育成ワークショップ
開催日時　：1月10日(金)
開催場所　：日本規格化協会(東京都新宿区)
受講料　　：10,000円
問い合わせ先：組込みソフトウェア管理者・技術者研究会(SESSAME), sessame@blues.tqm.t.u-tokyo.ac.jp
http://blues.tqm.t.u-tokyo.ac.jp/esw/opensessame/

ERモデルによるデータベース設計技法～基礎から実践演習まで～
開催日時　：1月15日(水)～1月16日(木)
開催場所　：SRCセミナールーム(東京都高田馬場)
受講料　　：79,600円
問い合わせ先：(株)ソフト・リサーチ・センター, ☎(03)5272-6071
http://www.src-j.com/seminar_no/23/23_009.htm

ステートマシンの設計技術
開催日時　：1月17日(金)～1月18日(土)
開催場所　：CQ出版セミナールーム
受講料　　：25,000円
問い合わせ先：エレクトロニクス・セミナー事務局, ☎(03)5395-2125

無線LANの動向と新技術
開催日時　：1月20日(月)
開催場所　：オームビル(東京都千代田区)
受講料　　：49,500円(1口で1社3名まで受講可)
問い合わせ先：(株)トリケップス, ☎(03)3294-2547, FAX(03)3293-5831
http://www.catnet.ne.jp/triceps/sem/c030120n.htm

画像処理システム開発技術
開催日時　：1月21日(火)～1月23日(木)
開催場所　：高度ポリテクセンター(千葉県千葉市)
受講料　　：30,000円
問い合わせ先：雇用・能力開発機構 高度ポリテクセンター事業課, ☎(043)296-2582 http://www.apc.ehdo.go.jp/seminar/jyohousub_out/syousai/02IIW22053.html

Windowsリアルタイム拡張による計測・制御
開催日時　：1月22日(水)～1月23日(木)
開催場所　：高度ポリテクセンター(千葉県千葉市)
受講料　　：35,000円
問い合わせ先：雇用・能力開発機構 高度ポリテクセンター事業課, ☎(043)296-2582 http://www.apc.ehdo.go.jp/seminar/jyohousub_out/syousai/02IIW22082.html

はじめてのHDL
開催日時　：1月23日(木)
開催場所　：CQ出版セミナールーム
受講料　　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局, ☎(03)5395-2125

車載用半導体デバイスの技術開発動向と信頼性技術
開催日時　：1月24日(金)
開催場所　：オームビル(東京都千代田区)
受講料　　：52,500円(1口で1社3名まで受講可)
問い合わせ先：(株)トリケップス, ☎(03)3294-2547, FAX(03)3293-5831
http://www.catnet.ne.jp/triceps/sem/c030124n.htm

DC-DCコンバータ設計の基礎
開催日時　：1月30日(木)
開催場所　：CQ出版セミナールーム
受講料　　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局, ☎(03)5395-2125

オシロスコープ入門(1日コース)
開催日時　：1月30日(木)～1月31日(金)
開催場所　：日本テクトロニクス(東京都品川区)
受講料　　：20,000円
問い合わせ先：日本テクトロニクススクール窓口, ☎(03)3448-3015
http://www.tektronix.co.jp/News/School/main.html

無線LAN構築・運用のトラブルシューティング
開催日時　：1月30日(木)～1月31日(金)
開催場所　：SRCセミナールーム(東京都高田馬場)
受講料　　：76,000円
問い合わせ先：(株)ソフト・リサーチ・センター, ☎(03)5272-6071
http://www.src-j.com/seminar_no/23/23_019.htm

半導体製造装置入門
開催日時　：1月31日(金)
開催場所　：CQ出版セミナールーム
受講料　　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局, ☎(03)5395-2125

リアルタイムOSの基礎
開催日時　：2月6日(木)
開催場所　：CQ出版セミナールーム
受講料　　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局, ☎(03)5395-2125

Linuxデバイスドライバ開発入門
開催日時　：2月7日(金)
開催場所　：CQ出版セミナールーム
受講料　　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局, ☎(03)5395-2125

日程はすべて予定です. 問い合わせ先にご確認のうえ, お出かけください.

読者の広場

2002年12月号特集「多国語文字コード処理＆国際化の基礎と実際」に関して

▷今回はI/F誌に期待している内容からはかけ離れていましたが，他誌では取り上げられそうにないので，役に立ちました．SWIGの記事も参考になりました．（arts）

▷国際化にはいろいろと取り決めなければならない問題が多くてたいへんだということがよくわかりました．また，Linuxにおいては，日本の方々が大勢努力していらっしゃることに驚きました．（玉出のタマ）

▷今回，MS-DOS時代に経験した文字コード以来，このような領域は初めて見たという状態です．Windowsになってからは深く経験していないので，今回の特集で興味をもったというのが本音です．文字コードの変換という意味で感動したといえば，10年近く前に中国へ出張した際，現地の電子タイプライタの文字変換を発音表記で入力し，漢字に変換する方法をとっていると聞き，なるほどと思ったことを思い出します．（匿名希望）

▷いつも悩まされる文字コードの特集を，ここまでよくまとめたなぁというのが率直な感想です．Mookとして出版してもらっても良いのではないかと思います．（JR9JUK）

▷特集は，多くのトピックについて触れられており，読み物として楽しめたし，今後の参考になると思う．また，多くの参考資料がURIで紹介されたのは時代の流れかとも思った．個人的にはLi18nuxにおける既存ソース/ライブラリとの運用上の注意点などを実例を含めてまとめてあればと思った．今後の記事に期待します．（信）

▷ソフトウェアの国際化ですが，日本が対応をせまられるばかりではつまりません．逆に自信のあるソフトウェアは，インターネットからアクセスできなくし，日本語版のみしか作らないという具合にできないでしょうか．「欲しかったら出すもの出して買いに来い」というわけです．記事とは相反するようですが，これぐらいのことがいえるソフトウェア（ハードウェア？）を作ってみたいものです．（H²）

▷コード系の特集は良いのですが，実際，Windowsなどの各種ブラウザでは，日本語ファイル名のファイルがダウンロードできないので，使い勝手が悪い．なぜ，ワールドワイドなのに，そのようなことが起こるのか，今後どうなるのかも含めて特集してほしかった．（てんりん）

その他

▷最近，組み込みシステムの仕事からは離れていますが，この分野についての情報はI/F誌だけなので，毎月購読しています．「開発者のためのアセンブラ入門」と「組み込みプログラミングノウハウ入門」は，プログラミング入門シリーズの中に加えてください．（帰ってきた青年実業家）

▷『ネットワーク向け3D規格「XVL」の概要』は，とても勉強になりました．ネットワーク向けの3D規格があるのさえ知りませんでした．Webはまだまだ2Dなので，3Dで物の形状がわかるようになると良いですね．（福沢陽介）

▷ずっとプログラム作成から離れていたので，なかなか昔の勘が戻ってきません．読んでいてもまだまだわからないことが多いのですが，少しずつ前に進んでいます．客先のプレゼンを行ううえで，XVLの記事はとても参考になりました．設計・製造のレベルだけに使うのみではなく，客先の購入意欲をそそるためにも十分に有効なツールだと思います．（まだ浦島太郎）

▷「ハッカーの常識的見聞録」は，毎月読んでいて，「よくもまぁこれだけ」と，妙に感心してしまいます．しかし，これが本当に「常識」なんでしょうか？（観空）

特集担当デスクから

☆何年前だったでしょうか，無線LAN対応機器を使ったインテグレーションをやっておられる会社に伺って打ち合わせさせていただいたのは．そのときは，無線LANを工場内の自走システムなどに積み，通信させて……といった話で，異機種との相互接続性などは，有線LANではあったけれど無線では課題ですねぇ，といった感じでした．ところがあっという間に，ADSLを代表とするブロードバンド化の流れとともに，IEEE 802.11b対応ワイヤレスルータなどが，家庭にまで入り込んできました．ベンダの方とお話しすると，相互接続もおおむね問題ないといわれます．

☆しかし，今月の特集第1章でも言及されていますが，無線LANについて，機器の設定などの情報は多くあるのに，技術や標準化動向などが解説されたドキュメントは意外と少ないというのが現状です．さまざまな可能性の拓ける分野でもあり，多くの技術者の興味をひけるのではと思って特集を企画しましたが，いかがだったでしょうか．

☆また，この分野の開発はどんどん加速化している感じがします．かなり普及してきたといえるIEEE802.11bでは最高伝送速度11Mbpsがうたわれていますが，54MbpsのIEEE802.11a対応機器が発売され，さらに100Mbps以上を実現するUWBや60GHz帯を使う伝送についても，さまざまな研究プロジェクトがたちあがっています．

読者の広場

アンケートの結果

興味のあった記事（2002年12月号で実施）

①第3章 文字コードの変換の実際
②第2章 多漢字問題とTRONコード
③第1章 文字コードの理論と実際
④第4章 Linuxにおける多国語対応の実際 —— I18N/L10N/G11N/M17N
⑤ネットワーク向け3D規格「XVL」の概要
⑥JPEG2000/Motion-JPEG2000の技術概要と応用(後編)
⑦第5章 Qtをサンプルとした国際化の実例
⑧フジワラヒロタツの現場検証(第65回)
⑨第6章 Windows CE .NETの国際化対応の概要
⑩フリーソフトウェア徹底活用講座(第4回)
⑪シニアエンジニアの技術草子(弐拾弐之段)
⑫Appendix ソフトウェア国際化プロセスの事例
⑫開発環境探訪(第14回)
⑭開発技術者のためのアセンブラ入門(第13回)
⑮組み込みプログラミングノウハウ入門(第7回)
⑮プロセッサコア「Xtensa」によるソフトウェア主体の設計手法(前編)
⑰ハッカーの常識的見聞録(第24回)
⑱第4回自動認識総合展
⑲NEW PRODUCTS
⑳Show & News Digest

特集「多国語文字コード処理&国際化の基礎と実際」についてのアンケートの結果

Q1　ふだんソフトウェアを作成するときに国際化を意識していますか？

①している（20%）
②していない（80%）

Q2　（Q1で①と答えた方にお伺いします）国際化ソフトウェアを作成するうえで，どのようなことに注意していますか？（複数回答可）

① Unicodeの使用（60%）
② ISO10646の使用（0%）
③ TRONコードの使用（20%）
④ロケールへの対応（20%）

Q3　国際化に関連した記事で，どのような記事をご希望ですか？

- 国際化を意識するとき，その地域でのソフトウェア環境が問題になると思います．したがって，EU，北米，アジアなど，地域的なソフトウェアの傾向についての記事があればよい
- 日本語の文化的な問題と，文字コードの整合性
- 世界中が欲しがるソフトウェア（ゲーム以外）の情報があまりないので，ぜひ取り上げてほしい．「ない」なら「ない」ではっきりと書いても良いと思う

2003年3月号は
1月25日発売です

次号予告

ICカード技術の基礎と応用

MULTOS/NICSS/eTRON/ASEPcos/Felica/JavaCard

これまで使用されてきた磁気カードや，本誌で解説を行ってきたメモリカードに加えて，ICカードが注目を集めている．単なる記憶装置にすぎなかった磁気カードやメモリカードと比べ，ICカードはその内部にCPUとメモリを搭載している．なおかつ，その中でOSを動作させたうえでセキュリティレイヤを搭載し，アプリケーションソフトウェアを動作させることにより高いセキュリティを実現している．また，ICカード用OSとしてもさまざまなものが発売され，さらに非接触カードと接触カードでは使用されるハードウェアが異なる．

そこで本特集では，ICカードアプリケーションを作成するにあたって必要とされる基礎知識から，ハードウェア，OS，アプリケーション，セキュリティの確保にいたるまでを徹底的に解説する．

編集後記

■ Embedded Technology 2002，反省点を残しつつも無事に終わりました．ビッグサイト(東京)からパシフィコ(横浜)に移り，皆さん来ていただけるか少々心配だったのが，蓋を開けてみると多くの人に来場いただき，ありがとうございました．技術現場の人が直接会っていろいろ語れる「現実の」場の必要性は高まっていると思います．（洋）

■「風邪を引いて寝ていた」とは，よく使われる表現だが，正確にはちょっと違うと思う．厳密には「風邪を引いてしまったので，布団を敷いて寝ていた」が正しいのではないのだろうか．単なる言葉遊びと思われるかもしれないが，じつはその通り！　風邪で発熱中なので，この程度の事しか考えられないのだ．この季節，ご自愛あれ．（＝IO）

■モノクロ再生ビデオは前も書きましたが，最近はもう1台のビデオもイジェクトの調子が悪いわ中でテープをかんだりするわ……フタを開けてみても調整できそうな部分もなし……．R/RWかRAMかHDD，はたまた融合機を買うか……青いのはまだ先みたいだしなぁ～．メディアも悩むがチューナ側も悩み物……地上波ディジタルはいつ？（M）

■もともと貨物線だったところに埼京線を走らせて，湘南・新宿ラインを乗り入れたうえにりんかい線まで走らせますか……．複雑なダイヤを構築するだけでも大変なのに，日々これを運用していかなければならない，と考えると気が遠くなります．頑張ってください．（み）

■本誌の場合，「組み込み」あるいは「エンベデッド」がキーワードである．最初はインテルが言い始めたらしく，CPUを機械に埋め込むということから使われたようだ．もっとたくさんCPUを使ってくれ，ということだったのだが，今ではCPUのない製品の方が珍しい．最近は，PC以外の製品を組み込み機器と呼んでいるようで，少し変な感じがする．（Y）

■不況ということもあり，少しでも明るい気持ちになれるようにと，デパートやホテルでは11月上旬からクリスマスイルミネーションを飾り付けていました．TVで見るたびに「綺麗だな」とは思うのですが……．今年はイルミネーションを見ながら気持ちだけでも暖かい気分になりたいですね．（Y2）

■携帯電話の市場も価格とサービスの厳しい競争に入り各社とも最新製品発表が短い時間で行われています．特に，画像表示をする機能を重点的に提案しています．中でも，動画を利用できるサービスを提供している携帯もあり，自動車の最高峰F1の中にも採用されているようです．今後は，身の回りのセキュリティを含めて利用範囲が広がり，数十年前のアニメに登場したような環境になるかもしれません．（S）

■親しらずを抜いた方がいいと，たまたま行くことになった大学付属病院で言われました．痛いわけじゃ無いんですよ！　歯茎から頭が出てきてもいないんです．でもその歯茎の中で止まっている状態が，かえって悪いらしい．親しらずの抜歯は未経験……．ただただ，脅えるばかりです．（A）

お知らせ

▶読者の広場
本誌に関するご意見・ご希望などを，綴じ込みのハガキでお寄せください．読者の広場への掲載分には粗品を進呈いたします．なお，掲載に際しては表現の一部を変更させていただくことがありますので，あらかじめご了承ください．

▶投稿歓迎
本誌に投稿をご希望の方は，連絡先(自宅/勤務先)を明記のうえ，テーマ，内容の概要をレポート用紙1～2枚にまとめて「Interface投稿係」までご送付ください．メールでお送りいただいても結構です(送り先はsupportinter@cqpub.co.jpまで)．追って採否をお知らせいたします．なお，採用分には小社規定の原稿料をお支払いいたします．

▶本誌掲載記事についてのご注意
本誌掲載記事には著作権があり，示されている技術には工業所有権が確立されている場合があります．したがって，個人で利用される場合以外は，所有者の許諾が必要です．また，掲載された回路，技術，プログラムなどを利用して生じたトラブルについては，小社ならびに著作権者は責任を負いかねますので，ご了承ください．
本誌掲載記事をCQ出版(株)の承諾なしに，書籍，雑誌，Webといった媒体の形態を問わず，転載，複写することを禁じます．

▶コピーサービスのご案内
本誌バックナンバーの掲載記事については，在庫(原則として24か月分)のないものに限りコピーサービスを行っています．コピー体裁は雑誌見開きの，複写機による白黒コピーです．なお，コピーの発送には多少時間がかかる場合があります．

- コピー料金(税込み)
  1ページにつき100円
- 発送手数料(判型に関わらず)
  1～10ページ：100円，11～30ページ：200円，31～50ページ：300円，51～100ページ：400円，101ページ以上：600円
- 送付金額の算出方法
  総ページ数×100円＋発送手数料
- 入金方法
  現金書留か郵便小為替による郵送
- 明記事項
  雑誌名，年月号，記事タイトル，開始ページ，総ページ数
- 宛て先
  〒170-8461　東京都豊島区巣鴨1-14-2
  CQ出版株式会社　コピーサービス係
  (TEL：03-5395-4211，FAX：03-5395-1642)

▶お問い合わせ先のご案内
- 在庫，バックナンバー，年間購読送付先変更に関して
  販売部：03-5395-2141
- 広告に関して
  広告部：03-5395-2133
- 雑誌本文に関して
  編集部：03-5395-2122

記事内容に関するご質問は，返信用封筒を同封して編集部宛てに郵送してくださるようお願いいたします．筆者に回送してお答えいたします．

Interface
©CQ出版(株)　2003　振替　00100-7-10665
2003年2月号　第29巻　第2号(通巻第308号)
2003年2月1日発行(毎月1日発行)
定価は裏表紙に表示してあります

発行人／蒲生良治
編集人／相原　洋
編集／大野典宏　村上真紀　山口光樹　小林由美子
デザイン・DTP／クニメディア株式会社
表紙デザイン／株式会社プランニング・ロケッツ
本文イラスト／森　祐子　唐沢睦子
広告／澤辺　彰　中元正夫　渡部真美

発行所／CQ出版株式会社　〒170-8461　東京都豊島区巣鴨1-14-2
電話／編集部(03)5395-2122　URL http://www.cqpub.co.jp/interface/
広告部(03)5395-2133　インターフェース編集部へのメール
販売部(03)5395-2141　supportinter@cqpub.co.jp
CQ Publishing Co.,Ltd.／1-14-2 Sugamo, Toshima-ku, Tokyo 170-8461, Japan
印刷／クニメディア株式会社　美和印刷株式会社
製本／星野製本株式会社

日本ABC協会加盟誌
(新聞雑誌部数公査機構)　ISSN0387-9569

Printed in Japan